ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA (ICAI)

BEHAVIOR DETECTION MODELS USING COMPUTER VISION APPLIED TO SECURITY SYSTEMS

Autor: Manuel Alvar Miró

Directores: Álvaro Sánchez Miralles

José María Armingol Moreno

Madrid
October 2014

Resumen

Esta tesis propone un modelo completo para la detección de comportamientos anómalos usando técnicas de Visión por Computador. Como resultado de ella se presenta el modelo básico de todo sistema de seguridad enfocado a la detección de anomalías compuesto por tres grandes bloques: detección de movimiento, seguimiento de objetos interesantes y análisis del comportamiento.

El trabajo de investigación comienza abordando el problema de la detección de movimiento. Se revisan los algoritmos más ampliamente usados en la literatura técnica analizando los más interesantes, así como sus ventajas e inconvenientes. A continuación se proponen dos modelos originales para la detección de movimiento de manera eficiente, centrados en la reducción del tiempo de cálculo consumido: los modelos **SDGM** y **MMGA**. El primero, partiendo de la información de dónde se encuentran las entradas y salidas de la escena analizada, reduce ampliamente el espacio de búsqueda a las zonas de interés. El segundo, a través de la aplicación innovadora del proceso de actualización del RTDENN a los modelos de sustracción de fondo, disminuye el tiempo de actualización considerablemente. Estas reducciones de tiempo se consiguen sin perder precisión en la detección, al mantener constante la tasa de fallos. El tiempo ahorrado gracias a dichos modelos está disponible para el modelo de análisis del comportamiento, permitiendo así que el modelo completo de detección de anomalías se pueda ejecutar en tiempo real, esencial en una aplicación de vigilancia o seguridad.

La salida del modelo de detección de movimiento se usa en el apartado de seguimiento. Cada conjunto de píxeles, denominados *blobs*, debe ser analizado para obtener de manera robusta las trayectorias de los diferentes objetos y personas que han atravesado la escena bajo vigilancia. Por una parte, el algoritmo propuesto construye unos histogramas que almacenan la información sobre los colores de los cuales está compuesto cada blob. Por otra parte, se descomponen los posibles problemas a afrontar en seis casos. De esta manera, se automatiza la solución de cada uno de ellos, permitiendo una resolución rápida. Los casos se diferencian en el número de blobs existentes en el fotograma actual, y el número de objetos que se habían detectado en el fotograma anterior. Tanto los unos como los otros se casan entre sí usando la información guardada en los histogramas mencionados, obteniendo así todos los puntos por donde cada objeto ha pasado. Los problemas de oclusiones se resuelven guardando la información de los objetos ocluidos por otros objetos, de manera que

pueda ser usada más adelante cuando vuelvan a aparecer los objectos ocluidos. El algoritmo permite, pues, construir la trayectoria de los objetos visibles, y una red de posibles trayectorias con diferentes probabilidades para cada objeto ocluido.

Por último, la tesis presenta un modelo de detección de anomalías usando la información recogida durante el seguimiento de personas. A partir de una muestra de trayectorias no supervisadas, el modelo se entrena para formar los conjuntos de comportamientos considerados normales. Por primera vez se usan los *Dominant Sets* (DS) como técnica de creación de conjuntos en una aplicación de análisis del comportamiento, permitiendo crear un sistema de seguridad de fácil instalación. Una vez entrenado, el modelo es capaz de detectar qué comportamientos son normales, y hacer saltar una alarma en caso de observar un comportamiento desconocido. Los resultados obtenidos muestran que la aplicación de DS a un sistema de vigilancia supera otros métodos para la detección de anomalías.

Abstract

This thesis proposes a complete abnormal behavior detection model using computer vision techniques. The thesis presents the basic steps of every security system model aimed at detecting anomalies: motion detection, interesting objects tracking and behavior analysis.

First, the thesis tackles the problem of motion detection. It reviews the algorithms most used in the technical literature and it details the advantages and disadvantages of each of these algorithms. Subsequently, two original models to efficiently detect movement in the scene are presented: the **SDGM** and the **MMGA**. Both models focus on reducing time consumption. The former, based on the location of the entrances and exits to the scene, greatly reduces the search space to certain regions of interest (ROI). The latter, using a novel application of the updating process of the RTDENN to the background subtraction models, considerably diminishes the time needed for the update of the parameters. These reductions in time are reached without losing accuracy, as the error rate remains stable. The time saved thanks to the models presented is available for the behavior analysis model, and thus allows the abnormal behavior detection model to work in real time, an essential characteristic of any security or surveillance application.

The object tracking step uses the output of the motion detection model. The system has to analyze each group of pixels, called *blobs*, in order to robustly obtain the trajectories of the different objects and people crossing the scene under surveillance. On the one hand, the proposed algorithm builds histograms that store the color information of the pixel a blob is composed of. On the other hand, it decomposes any problem a tracking system may confront into six different cases. In this way, the tracking step automatizes the solution of these easily, and permits a fast resolution of the tracking step. The number of blobs appearing in the present frame and the number of objects detected in the previous frame differentiate one case from the others. The model matches present blobs and previous objects using the information stored in the color histograms and, point by point, builds the trajectory each object has taken. Occlusion problems are solved saving the information of occluded objects under others, in order to use it in the future when the occluded object is visible again. Thus, the algorithm allows the construction of the trajectory of visible objects, and a set of trajectories, each with a different probability, for each occluded object.

Finally, the thesis presents an abnormal behavior detection model that uses the information gathered during the people tracking step. The model is trained using an unsupervised training set of trajectories, forming a set of clusters that model the behaviors considered as normal. For the first time, *Dominant Sets* (DS) is applied as the clusterization method in a behavior analysis model. Taking into account that DS does not need any supervision, any system using the presented behavior analysis model is an easy-to-install system, ideal for security applications. Once trained, the model detects the normal behavior, and triggers an alarm in cases where the behavior is unknown. The obtained results show that the application of DS in a surveillance system outperforms other abnormality detection models.

Contents

List of Tables

List of Figures

Nomenclature

α	Weight learning rate of MGM.
α_{NG}	Value of learning rate α for pixels not belonging to a gate.
β	Significance level of the Chi-Square test.
$\boldsymbol{\mu}_{i,N}$	Mean of the Gaussian i containing the last N.
$\boldsymbol{\mu}_{k,t}$	Mean of the kth Gaussian/neuron at time t.
$\boldsymbol{\mu}_k$	Mean of values belonging to R_k.
$\boldsymbol{\mu}_N$	Mean of the new set of N values in RTDENN.
$\boldsymbol{\Sigma}_{i,N}$	Covariance of the Gaussian i containing the last N.
$\boldsymbol{\Sigma}_{k,t}$	Covariance matrix of the kth Gaussian/neuron at time t.
$\boldsymbol{\Sigma}_k$	Covariance matrix of values belonging to R_k.
$\boldsymbol{\Sigma}_N$	Covariance matrix of the new set of N values in RTDENN.
$\boldsymbol{I}$	Identity matrix.
$\boldsymbol{X}_t$	Value of the pixel in frame t.
δP_{it}	Boundary of polygon P_{it}.
Δ_n	Standard simplex of $\mathbb{R}^n$.
$\hat{P}_{i(t+1)}$	Polygon corresponding to the estimated position of moving object i in frame $t+1$.
λ	Coefficient to determine the maximum distance of a pixel value to the mean of the Gaussian distribution so that the pixel fits in.
μ_k	Mean of dimension k values.
μ_{k^2}	Mean of the squared values of the dimension k.
$\mu_{x,k,t}$	Value of element (x) of $\boldsymbol{\mu}_{k,t}$.

$\phi_S(i,j)$	Measure of the similarity between nodes j and i with respect to the average similarity between node i and its neighbors in S.
ρ	Mean learning rate in MGM.
ρ_{NG}	Value of learning rate ρ for pixels not belonging to a gate.
$\|S\|$	Cardinality of S.
$\sigma(\boldsymbol{x})$	Support of $\boldsymbol{x}$, set of indices corresponding to its positive components.
σ_k^2	Variation of dimension k values.
$\sigma_{k,t}$	Standard deviation of the kth Gaussian/neuron at time t.
$\sigma^2_{x,k,t}$	Value of element (x,x) of $\boldsymbol{\Sigma}_{k,t}$.
$\sigma_{x,y,k,t}$	Value of element (x,y) of $\boldsymbol{\Sigma}_{k,t}$.
$\overset{*}{P}_{it}$	Region inside polygon P_{it}.
$\mathbf{v}$	Estimation of the movement vector of the polygon centroid of a Dynamic Gate.
$\boldsymbol{e}_i$	i-th vector of the canonical base, i.e., a vector having i-th component equal to 1 and all other components equal to 0.
$\boldsymbol{x}^S$	Characteristic vector of S.
$\vec{F}(i)$	Normalized 4D vector of element i element of data set.
$\widetilde{w}_{k,t}$	Normalized weight of the kth Gaussian at time t.
A	Similarity matrix.
a_{ij}	Similarity between node i and node j of V.
$awdeg_S(i)$	The (average) weighted degree of object i of V with respect to S.
B	Position in the Gaussians' ordered list of the last Gaussian to be considered to be modeling background values.
b	A blob.
$BC(h^1,h^2)$	Bhattacharyya coefficient.
BC_C	Bhattacharyya coefficient between color histograms.
BC_G	Bhattacharyya coefficient between grayscale histograms.
c	decreasing rate coefficient of Dominant Sets model.

$d(b,o)$	Hellinger distance between blob b and object o.
$d^*_{i,k}$	Normalized value of dimension k of i^{th} vector.
$d_{i,j}$	Distance between Gaussian/neuron i and j.
$d_{i,k}$	Value of dimension k of the i^{th} input vector.
D_{te}	Size of test data set.
D_{tr}	Size of training data set.
DG_{it}	Dynamic Gate i at time t.
$Encrec$	Enclosing rectangle of a blob.
F	Update rate in SDGM.
g_i	Gaussian i.
G_{it}	Gate composed of the pixels *inside* $\overset{*}{P}_{it}$.
$h(k)$	Value of k^{th} bin of histogram h.
H_{convex}	Convex hull of a blob.
I_t	Image at time t.
K	Number of Gaussians in MGM, SDGM or MMGA.
$L_{k,t}$	Binary variable which equals to 1 when the new pixel fits the kth Gaussian.
M	Dimension of space R of the RTDENN model.
N	Number of frames for the update of the background model in MMGA.
$N_S^{traj}(o,j)$	Total number of segments belonging to trajectory j of object o.
N_P	Number of polygons.
n_k	Number of samples that belong to the region R_k.
$N_S(o,t)$	Total number of segments object o may be in at time t.
nn_k	Neuron k, specialized in subspace R_k.
o	An object.
p	A pixel.
$P(o,s,t)$	Probability of object o being in segment s at time t.
$P^{dis}(o,t)$	Probability of object o having disappeared at time t.

$P^{inh}(O_n, O_o)$ Probability of object O_n inheriting object O_o trajectories.

$P_o^{traj}(j)$ Probability of trajectory j of object o.

$P_{acc}^{dis}(o,t)$ Probability of object o having disappeared before time t.

P_{it} Convex polygon i at time t.

p_{it}^{j} Pixel/point belonging to P_{it}.

R M-dimensional space used in RTDENN model.

R_k Subspace of R.

S Subset of V.

s_i Segment i.

$SA(DG_{it})$ Resulting blob after the execution of the subtraction algorithm on gate DG_{it}.

SG_{it} Static Gate i at time t.

T_B Minimum percentage (threshold) of the total data analyzed considered to be background.

T_d Maximum acceptable color distance threshold.

V Set of nodes to be clustered.

$W(S)$ The total weight of S.

$w_S(i)$ The weight of i with respect to S.

$w_{k,t}$ Weight of the kth Gaussian at time t.

$X_{m,t}$ Value of channel m of vector $\boldsymbol{X}_t$.

Chapter 1

Introduction

1.1 Surveillance and security systems using computer vision

The aim of the visual analysis of the movement of people and objects is to detect, track and identify people, and overall, to interpret human behaviors. The interest in this field lies in the promising applications of computer vision in several domains, such as security. Nowadays, this field, and even more the application of computer vision techniques to the special case of surveillance systems treated here, is one of the most active research topics in computer vision [Wang et al., 2003].

The main goal of this thesis work is to develop a smart system capable of detecting abnormal situations and of helping security operators to correctly analyze the monitored scene. Nowadays, video processing is broadly used for surveillance systems, as well as for many different applications, in view of the large volume of information it can provide, its low cost and its robustness [Regazzoni et al., 2001]. Visual surveillance research, a specific research area of the wider security system research field, focuses on the development of artificial intelligence systems that improve traditional passive security systems [Ko, 2008]. [Hampapur et al., 2003] proved that such passive systems are inefficient as the number of cameras and the amount of information exceed the human operator's ability to monitor them. Initially, digital video surveillance systems were used exclusively to capture, store and distribute video, while analysis of the video was carried out by human operators [Jan, 2004]. This stored information was used *a posteriori*, after it was known an interesting event had happened. A modern security system, and more specifically, a modern surveillance system not only aims to install cameras in places to monitor and substitute the human eye, but also to carry out the surveillance automatically and autonomously. Nowadays, smart surveillance systems are increasingly being used to set alarms in potentially dangerous situations as a support to human operators, providing useful information and facilitating their task

by interpreting the environment [Dick and Brooks, 2003]. To achieve this, there have been improvements in the hardware structure used in surveillance and security systems. Hampapur et al. present three evolutions of a traditional architecture in [Hampapur et al., 2003]: from a Basic Smart Surveillance Architecture to a modern Distributed Smart Surveillance Architecture. The same authors also enumerate different types of alarms that those system could generate [Connell et al., 2004]. The alarms in a system would depend on the area guarded, the position of the cameras and on user needs.

Figure 1.1. Example of a surveillance control center.

Recently, several leading international journals have published articles concerning human activity analysis using computer vision, such as “IEEE Transactions on Pattern Recognition and Machine Intelligence”, “International Journal of Computer Vision”, “Image and Vision Computing”, “Computer Vision and Image Understanding”, “IEEE Transactions on Image Processing” or “Computer Vision and Image Understanding”. Moreover, numerous prestigious conferences have addressed different problems in this area of research at meetings such as the “International Conference on Computer Vision”, the “IEEE International Workshop on Performance Evaluation of Tracking and Surveillance”, the “IEEE International Conference on Computer Vision and Pattern Recognition”, the “IEEE International Conference on Advanced Video and Signal based Surveillance”, the “TREC Video Retrieval Evaluation” or the “International Conference on Pattern Recognition”. This interest in the field herein dealt with shows the current utility of and need for research into autonomous security systems based on computer vision techniques.

In the technical literature, there is agreement on the basic steps to follow in order to obtain a reasonable analysis of the human activity required to trigger, or not trigger, alarms. The common basic steps to achieve this task in a computer vision-based system are motion detection, interesting object tracking and behavior analysis [Wang et al., 2003, Hu et al., 2004a, Connell et al., 2004, Ko, 2008].

The basic structure of such systems begins with detection of the movement of the interesting object. The system has to model the scene being filmed by the camera so that movements can be robustly detected. The motion detection step creates a binary

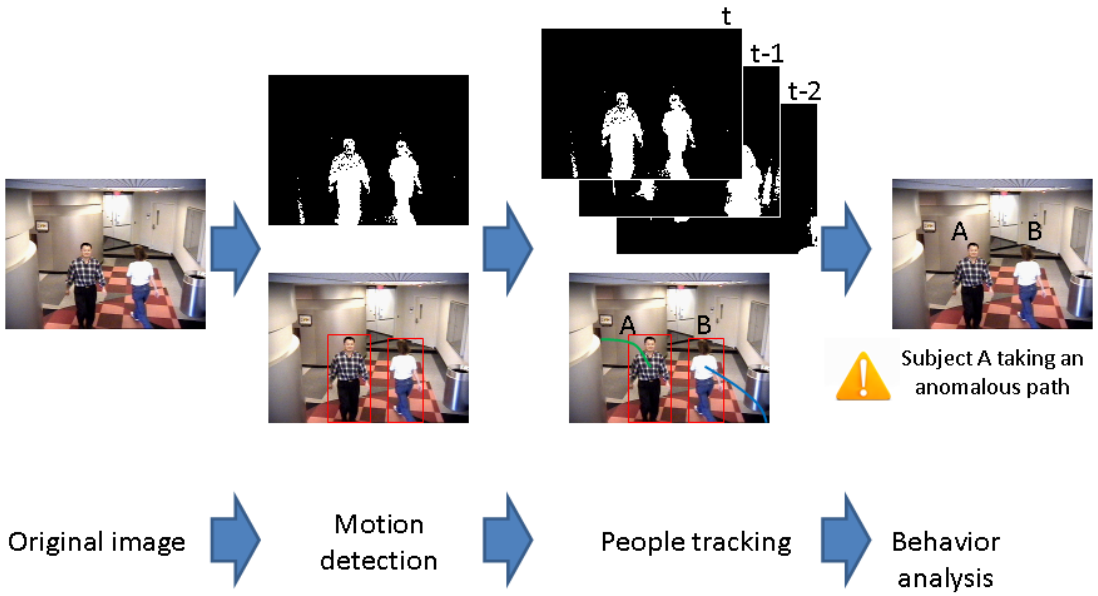


Figure 1.2. Structure of a security system.

image where white pixels correspond to the pixels where there is movement in the current frame, and black pixels if there is not. Then blobs, composed of the set of white pixels close to each other, are created. These blobs can be a person, a group of person or a limb [Zhao and Nevatia, 2004], and thus they might require a further analysis.

The tracking algorithm is in charge to carry out this analysis. The results of motion detection step, the blobs, are used as the tracking algorithm inputs to carry out the analysis. Moreover, the tracking step obtains the location of each object in every frame. Although motion detectors provide enough information to accomplish a basic behavior analysis, it is necessary to obtain the whole trajectory of each interesting object of the scene in order to robustly detect more complex abnormal behavior. That is, the tracking step has to obtain when and where each object has been.

Once tracking information has been gathered, behavior analysis and understanding is performed. The main objective, in a security system, and specifically in a surveillance system, is to detect abnormal situations to warn the operators and simplify their job. The behavior analysis models are trained offline and then used online to automatically detect anomalies in people behaviors. The behavior analysis model is capable to detect the anomalies in the values of the data obtained in previous steps, motion detection and object tracking in this case, and, unfortunately, only these anomalies.

These steps are thoroughly described in the following chapters 2, 3 and 4. The first two steps correspond to lower level image processing techniques, while behavior analysis is solved at a higher level through machine intelligence techniques.

1.2 State of the art

Motion detection, interesting objects tracking and behavior analysis are the three main research fields essential in a surveillance system based on computer vision. Several different techniques have been proposed in the technical literature aimed to solve these problems. This section presents a brief description of the most important and interesting techniques of each of these fields. A detailed revision of the state of the art of each of these field can be found in sections 2.2, 3.2 and 4.2 respectively.

1.2.1 Motion detection

Most of the motion detection models already published can be classified in four main techniques: frame difference, background subtraction, optical and pattern analysis.

Frame difference [Lipton et al., 1998] compares the value of each pixel in the present frame with the values in the previous. If the difference is wide enough, the algorithm consider there is motion in that specific pixel. It is a simple and fast calculation, and, taking into account that the only data the algorithm keeps from frame to frame are the last frame values, frame difference is the fastest technique to detect movement. However, it lacks the robustness of other techniques, as it fails to detect slow movement [Dockstader and Tekalp, 2000, Munkelt and Kirchner, 1996].

Background subtraction is an evolution from frame difference, where instead of keeping the last frame, it generates and updates a model, called background model, where each pixel models the value of the background. Background subtraction obtains largely better results than frame difference, with only a small reduction in time effectiveness. The main drawback of this technique, which is share with frame difference, is the necessity to work with static cameras [Yu et al., 2007], so that the background image correspond to the scene recorded. Wren et al. proposed to model each pixel as a simple Gaussian in [Wren et al., 1996]. This method was improved by MGM in [Stauffer and Grimson, 1999]. A mixture of Gaussians models each pixel, allowing more than one background color per pixel, essential in scenes with waving branches, opened/closed doors, ocean waves, etc. MGM has become a widely used technique and a general background subtraction comparison model.

Optical flow [Barron et al., 1994, Beauchemin and Barron, , Fleet and Weiss, 2006] is a widely known technique used is many different research fields in computer vision and for many different applications. It robustly estimates motion of the objects, even in moving-camera situation. To obtain such good results, it uses complex algorithms that can hardly run in real-time [Zhan et al., 2007, Ko, 2008, Maddalena and Petrosino, 2008, Munkelt and Kirchner, 1996].

Pattern analysis models, such as the model presented in [Viola et al., 2003, Viola et al.,

2005], are used to detect known patterns in the scene. To do so, they need to be trained and thus to acquire and classify training samples for every different environment where the technique is used. This means that for each application and camera position, a set of representative samples must be provided to make it work properly. This technique uses a complex algorithm and, besides the need for pattern creation, is not suitable for use in real-time applications.

Table 1.1 sums up the effectiveness of the main motion detection techniques in four categories: ability to work in real time, robustness of the technique, how easy it is to be used in different environments and if it is able to work with moving cameras. It can be seen that background subtraction is the most interesting technique to be used in a surveillance system. Even more, taking into account that many of the security system installed nowadays use static cameras, reducing the impact of its main drawback.

Technique	Real time	Robustness	Exportable	Moving cameras
Frame difference	🟢🟢	🔴🔴	🟢🟢	🔴🔴
Background subtraction	🟢	🟢	🟢🟢	🔴🔴
Optical flow	🔴🔴	🟢🟢	🟢🟢	🟢🟢
Pattern analysis	🔴	🟢🟢	🔴🔴	🟢🟢

Table 1.1. Pros (🟢) and cons (🔴) of main motion detection techniques

Piccardi reviews the most important background subtraction techniques and claims that the best results, with limited resources, are attained by the MGM [Piccardi, 2004]. Nevertheless, many authors have pointed out that MGM can be outperformed in computational time [Kim and Park, 2006, Cermak and Keyzer, 2007, Tsai and Lai, 2009]. SPecifically, they propose two means to improve MGM: reducing the search are to interesting zones and optimizing the slow updating process.

1.2.2 Object tracking

There are two stages to obtaining the whole trajectory of an interesting object to be tracked: deciding which features are used to model the tracked object and tracking the object thanks to these features. Scale Invariant Feature Transform (SIFT) [Lowe, 2004] is one of the most widely used feature in several applications, but has a strong drawback, it consumes too many computing resources. Bay et al. propose an improvement to speed it up called SURF [Bay et al., 2006], but still not being fast enough to work in real-time applications.

Among the most common feature used in surveillance and security applications are boundaries, optical flow and color [Ohta et al., 1980]. On the one hand Edge detectors, such as Canny detection [Canny, 1986], is a fast solution. Unfortunately, it cannot track non-rigid objects, such as people. On the other hand, active contour representation

techniques [Sun et al., 2011] track non-rigid objects, but increment the consumption of computing resources. As has already been commented, optical flow, such as Lucas and Kanade technique [Lucas and Kanade, 1981], has a poor time consumption performance compared to faster techniques. Color, however, is the most broadly used classifier, but is also the classifier that requires the least post-processing and thus is the ideal feature to use in the case where execution time consumption is a major constraint. The color information is generally stored in form of histograms, as in [Nummiaro et al., 2003].

Table 1.2 presents all the above features highlighting their strong points.

Feature	Speed	Track people
SIFT	🔴🔴	🟢
SURF	🔴	🟢
Edges	🟢🟢	🔴🔴
Active contour	🔴🔴	🟢🟢
Optical flow	🔴	🟢
Color	🟢	🟢

Table 1.2. Pros (🟢) and cons (🔴) of main features

Yilmaz presents the most interesting models to track objects in different applications [Yilmaz et al., 2006]: point tracking, kernel tracking and silhouette tracking. Point trackers [Veenman et al., 2001] efficiently tracks points. It is only suitable to be used in environment where tracked objects are too small, which is no usually the case in surveillance systems. Silhouette tracking [Cremers and Schnörr, 2003] estimates the real object shape in every frame. It makes intensive use of either edge detection, shape matching or contour evolution techniques.

Template based techniques are the main kernel tracking techniques. In a template matching system, the algorithm looks in the whole image for a region similar to the templates available. As explained previously, the main flaw of this technique is intense resource consumption, however some improvements have been presented, combining other techniques [Santner et al., 2010].

Comaniciu et al. present a mean-shift approach to reduce the search space and the computing time compared to a brute force template matching algorithm, and then use histograms as a matching feature [Comaniciu et al., 2003]. Nevertheless, mean-shift-based approaches may lose tracking due to occlusions [Khan et al., 2011].

[Tao et al., 2002] propose a new model to cope with drifting or mismatching due to similar appearances. The shape (boundaries), appearance (color) and motion (speed) of the foreground objects, obtained after a layering decomposition, are analyzed. However, although robust against stationary objects or close-by objects, it does not correctly track articulated objects such as people.

A summary of the tracking techniques is shown in table 1.3, where it can be seen that all of them have at least an insuperable flaw. It is thus recommended, for a surveillance application, to use a novel fast tracker capable to robustly track people, even in situations with occlusions, that employs color histograms as tracking feature.

Technique	Real time	Track people	Speed information	Exportable
Points	🟢🟢	🔴🔴	🟢	🟢
Kernel - Templates	🔴🔴	🟢🟢	🔴	🔴🔴
Kernel - Mean-Shift	🔴	🟢🟢	🟢	🟢
Kernel - Layering	🔴	🔴🔴	🟢🟢	🟢🟢
Silhouette - Shape	🔴🔴	🟢🟢	🔴	🔴

Table 1.3. Pros (🟢) and cons (🔴) of main tracking techniques

1.2.3 Behavior analysis

Behavior analysis step of a security system tries to match a previously learned pattern of activities with the present activities. One of the first techniques applied to behavior analysis is Dynamic Time Warping (DTW) [Bobick and Wilson, 1997]. DTW is able to match a movement to a pattern independently of the time. The pattern and the movement analyzed can have different speed, the only important factors are the temporal location and the order of the states. Patterns are also used by Chomat and Crowley, who generate movement templates by applying PCA to a set of spatio-temporal filters [Chomat and Crowley, 1998]. The main advantage of the templates is its simplicity. However, it needs a labeled training set as well as a occlusion-free environment to work correctly.

State machine techniques are used in behavior understanding to determine whether any reference sequence of images matches with the analyzed sequence or not. Wilson et al. [Wilson et al., 1997] use a finite state machine with only 3 states to model the behavior of a person telling a story. The state machine techniques are suited to knowing which action, of the different actions known to the program, the people are doing. Thus they require a huge amount of supervised data to model the algorithm correctly and is not adapted to recognize new actions or states.

Hidden Markov models (HMM) are a technique that requires two stages: training and classification. In general this technique outperforms other template techniques such as DTW. Oliver et al. use HMM to model human behaviors and interactions such as meetings or one person following another [Oliver et al., 2000]. Unfortunately, supervised training sets are not always available and they restrict the applications of the behavior detection model considerably.

In a surveillance system aimed to detect abnormal situations, the common solution, when there is no dataset available to do the training of the system, is to use an unsupervised clustering, where dense clusters correspond to normal behaviors whereas rare events are considered anomalous [Nayak et al., 2011]. A fuzzy K-means-based algorithm is presented in [Hu et al., 2006]. The clustering takes place in a two-step model, where trajectories are first clustered on a spatial basis and then each cluster in turn is sub-clustered on a temporal basis. Both clusterizations are performed with a fuzzy K-means algorithm. Saligrama and Chen introduce a mixture of the background subtraction and the optical flow of pixels in each spatio-temporal mask as feature [Saligrama and Chen, 2012]. They obtain a descriptor that, when combined with a trained Nearest Neighbors clustering algorithm, detects local anomalies. [Hu et al., 2004b] present a neural network where the input values are the location and speed of the tracked object in every frame as well as its size, and the number of output neurons corresponds to the number of behavior clusters. The distance from each neuron to new trajectories is calculated. If it exceeds a threshold, the new trajectory is considered abnormal. A drawback of these works is that the number of clusters/neurons is not estimated and has to be set manually.

Table 1.4 outlines the main pros and cons of the behavior analysis techniques, when applied to a surveillance system. The main characteristic searched in a smart modern abnormal behavior detector is that it helps the security operator. It must perform automatic and autonomous detections. It is important thus that the behavior analysis model do not need any external information or help, such as a supervised training or manual parameter adjustments.

1.2.4 Other research fields applied to surveillance systems

Besides these three main steps in security systems using computer vision techniques, some authors point out other interesting steps that might be carried out to obtain better results. In cases where the system manages various sensors, information fusion should be carried out. In [Cristani et al., 2010], authors carry out a complete review of the techniques to combine image information with other sensors, such a audio sensors, for a background subtraction application.. When the scene under surveillance contains more than one camera, Brémond et al. [Brémond et al., 2006] propose combining the data from the cameras in the tracking step. That way, 3D tracking can be performed, and the information obtained can be used as input in the behavior analysis step. This leads to another promising research area, which is people location in the scene. Location can be critical for the correct performance of a security system. Remanignino et al. [Remagnino et al., 1997, Remagnino et al., 1998a, Remagnino et al., 1998b] use a behavior analysis module that takes ground plane coordinates into account. On their part, Feldman and Weinshall [Feldman and Weinshall, 2008], although not locating the people in a 3D scene perfectly, are able to determine depth ordering using only 2 consecutive frames. Finally,

Technique	**Pros/Cons**
DTW	Speed independent
HMM	Better than DTW Need Training
Templates	Easy-to-use Classification of each template Occlusions
States machines	Flexible Environment information
Optical flow + KNN	No training needed Number of clusters to be estimated
Neural networks	Easy-to-get input values Number of neurons to be estimated
Fuzzy K-Means	Unsupervised training Number of clusters to be estimated

Table 1.4. Pros (🟢) and cons (🔴) of main behavior analysis techniques

another useful aid for these systems is identification through facial recognition [Steffens et al., 1998] or gait [Phillips et al., 2002].

1.3 Scope of the thesis

As has been noted, modern smart surveillance and security systems has to alert when suspicious or anomalous activities occur. In many cases, it is too difficult to explicitly define what all abnormal behaviors are. Furthermore, the system even has to handle events that have not previously occurred. For this reason, such systems usually only learn normal behaviors and consider any unknown behavior as abnormal [Jiang et al., 2011].

In this case, only behaviors that do not fulfill requirements calculable by means of the values obtained from detection and tracking models are considered anomalous. That is, behaviors are classified as abnormal when they are not similar to known behaviors, due to a change in speed, a change of trajectory, a change in direction , etc. In real-life applications, the presented systems can detect as anomalies behaviors such as people not stopping in the line of a supermarket, cars not stopping to take a parking lot ticket, people exiting a building through an emergency only exit, people entering through an exit door in theaters or museums, entering to a house through windows or climbing a fence, people approaching paintings too much in a museums, etc. However, the systems would be unable to detect suspicious behavior based on what people are looking at or in special cases such as a person following another person, people exchanging objects, etc.

On the one hand, surveillance systems have to operate in real time. Therefore, the algorithms of motion detection, tracking and behavior analysis models should have the lowest processing times possible. Several researchers have demonstrated that there is room to improve detection and tracking speed [Lee, 2005, Tsai and Lai, 2009]. This thesis proposes the use of focused analysis models, thereby saving time as the search for motion is restricted to interesting regions. These regions of interest can be obtained by taking into account the past movements of each object, that is, using the tracking output as an input in the detection step. Moreover, an intelligent update model can also reduce drastically the computing time needed without incurring in a higher error ratio.

On the other hand, behavior analysis and understanding models support their proper functioning in a set of stored patterns with which they are trained [Bobick and Wilson, 1997, Bobick and Davis, 2001, Wang et al., 2003, Hu et al., 2004a]. This approach has a huge drawback: it cannot detect new behaviors. That is why it is important to develop models that can generate new online normal behavior patterns, similar to Bregler's work in [Bregler, 1997]. The problem in these models is the computational cost, which is still too high to run in real time (see [Aggarwal and Ryoo, 2011]). There is still room for development of new behavior description models, especially for models focused on real-time pattern creation.

A set of models for behavior analysis focused on the use in surveillance systems based on computer vision are presented in this thesis. The models have been developed as independent modules, so that each application can choose to implement the models that are interesting and replace the others by other motion detection, object tracking or behavior analysis models.

1.4 Thesis objectives

The aim of this thesis dissertation is to develop all the necessary modules in a complete smart surveillance system. Surveillance systems are the security systems aimed to detect abnormal human behaviors and help the operator to recognize dangerous situations. This main objective requires the development of several models to a) detect motion in the monitored scene, b) track interesting objects detected and c) analyze the behavior of such objects. In addition to the previous general objective, the specific objectives tackled in the research study of this thesis can be summarized as follows:

- Develop adaptive human behavior detection models using computer vision techniques. These models will subsequently be applied to the field of security systems to test their effectiveness. That is why the capability to run the model in real-time is a strong requisite.
- Develop algorithms to classify the different types of behavior to detect abnormal situations. This implies the development of a decision system that, once human behaviors have been estimated, is able to distinguish and classify them in order to determine if the analyzed scene represents an abnormal situation, with an error rate lower than or equal to that of the current behavior classification systems.

The following objectives also have to be fulfilled to ensure that the models developed in this thesis are viable in real surveillance scenes:

- Improve motion detection and tracking models. Both motion detection and tracking are indispensable for behavior understanding models, as they are previously executed to know who is in the scene and what they are doing. Improvements in time are mandatory to fulfill the requirements of this thesis.
- Find or create several comparison videos to test the performance of the different models created against other state-of-the-art techniques. The chosen videos have to fulfill certain characteristics to be acceptable as a comparison dataset. The motion detection performance evaluation needs a complete set of videos containing shadows, background motion, “ghosts” and stationary people. In the tracking step, desirable videos should contain partial and total occlusions, several

people crossing each other and people leaving the scene from several different points of the image. These two steps may use the same dataset if it contains videos that fulfill all the requirements. Behavior performance analysis demands long trajectories, where the normal behavior is repeated several times, while the abnormal behavior copes with all the different abnormalities the model should detect, such as different paths, different speeds or different headings in a similar trajectory. The dataset used in each step has to contain videos filmed in both outdoor and indoor environments. Additionally, if the ground truths of those videos are not available, they have to be created manually before developing the model to obtain a fair comparison. All the newly created videos and ground truth data have to be publicly available in order to enable other researchers to compare their techniques using the same video datasets.

1.5 Outline of the document

In order to cover all the previously described objectives this thesis dissertation is structured in five chapters. This section outlines and summarizes each of these chapters.

Chapter 2 presents the motion detection models developed in this work. First, after an introduction of the main motion detection techniques, this chapter focuses on background subtraction techniques in general and on the mixture of Gaussians model (MGM) in particular. In addition, it describes the recently presented Real-Time Dynamic Ellipsoidal Neural Network (RTDENN), originally used in robotics, and its learning processes. Secondly, the two novel models herein developed - Static and Dynamic Gates Model (SDGM) and Mixture of Merged Gaussians Algorithm (MMGA) - are presented. Both models have been tested in worldwide used motion detection datasets.

The new tracking model resulting from this thesis is introduced in **Chapter 3**, which presents a classification of any tracking process problem in 6 different cases to tackle any tracking challenge in a simpler and more efficient way. Moreover, this chapter introduces the concept of *segments* to help cope with occlusion problems and allow the tracking model to handle different possible trajectories, as well as the probability of each one occurring, for each occluded object in the scene.

Chapter 4 sets out the new behavior analysis and anomalies detector model developed in this thesis to cope with the objectives described below. It also provides an introduction to the Dominant Sets (DS) clusterization method along with this method's application to the problem of abnormal behavior detection. The model is compared to usual techniques and validated in several different situations

Finally, **Chapter 5** summarizes the conclusions, developments, and results that have been achieved in this thesis. It also states the main original contributions that this work

has yielded, and finally, suggests new lines of research.

Chapter 2

Motion Detection

2.1 Introduction

The smooth running of current security systems based on computer vision depends strongly on the correct interaction between the different algorithms it consists of. Usually, the first step of the application is motion detection in the scene [Howarth, 2005, Del Rose and Wagner, 2012]. The results of this step are the inputs of the subsequent algorithms. Therefore, motion detection is critical in this type of application in order to achieve a good performance [Zhang et al., 2009]. It is important both to have a robust algorithm of motion detection and to run it quickly enough to meet real-time application requirements.

Security applications based on computer vision require motion detection, which must deal with many challenging events. These systems should be able to deal with changing illumination, moving background objects, occlusions, left objects, shadows, etc. The camera may also introduce extra noise, causing pixels to vary from frame to frame, even though there is no movement in the scene. Furthermore, these applications should run in real time.

There are many motion detection techniques referenced in the technical literature, of which background subtraction is one of the most commonly used. This chapter presents two models to speed up the background subtraction algorithm. The mixture of Gaussians model (MGM) proposed by Stauffer and Grimson [Stauffer and Grimson, 1999] is a common solution used to solve this kind of problem and has, therefore, been chosen as the base algorithm for the proposed models. This chapter will show how to improve MGM by making it faster while maintaining an error ratio similar to that of MGM used alone. First, motion detection focuses on interesting regions called gates [Alvar et al., 2014a]. Secondly, an improvement to the update mechanism is proposed [Alvar et al., 2013] that uses the learning process of the Real-Time Dynamic Ellipsoidal Neural Network (RTDENN) model [Sánchez Miralles and Sanz Bobi, 2002,

Sánchez Miralles and Sanz Bobi, 2004, Sánchez Miralles and Sanz Bobi, 2006].

The chapter is organized as follows: Section 2.2 gives an overview of and discusses previous research. A brief explanation of the MGM will be given in Section 2.3, while Section 2.4 deals with the RTDENN model. The proposed models will be explained in Sections 2.5 and 2.6 respectively. Section 2.7 presents the experiments and the results, which are discussed in Section 2.8, the conclusion.

2.2 Previous works

Previous works have implemented different approaches to solving the motion detection problem in security systems. This section reviews the main motion detection techniques, showing that background subtraction models are a suitable and standard solution to the problem. It also illustrates how the technical literature has already proposed focusing on some restricted areas, with improved results. Finally, Subsection 2.2.4 studies some of the algorithms that combine motion detection and tracking.

2.2.1 Main motion detection techniques

The main motion detection techniques are pattern analysis, optical flow, frame difference and background subtraction [Ko, 2008]. Zhan et al. [Zhan et al., 2007] explain the differences between these methods. First, pattern analysis, such as the model presented in [Viola et al., 2003, Viola et al., 2005], needs training samples. This means that for each application and camera position, a set of representative samples must be provided to make it work properly. This technique uses a complex algorithm and, besides the need for pattern creation, is not suitable for use in real-time applications. Secondly, optical flow is a technique that also uses complex algorithms that can hardly run in real-time [Zhan et al., 2007, Ko, 2008, Maddalena and Petrosino, 2008, Munkelt and Kirchner, 1996]. However, Li and Wang propose a new formulation using neural computing to parallelize processing and achieve better computation times [Li and Wang, 1993]. Frame difference [Lipton et al., 1998], although using a simple and fast algorithm, is not an accurate method to detect motion in a scene. Dockstader and Tekalp [Dockstader and Tekalp, 2000] explain that in spite of being an efficient technique, it is not suitable for some situations, such as moving objects that stop their movement [Munkelt and Kirchner, 1996]. Background subtraction can be faster than most models, but it is only applicable in scenes with static cameras [Yu et al., 2007] and therefore, unacceptable for applications such as those in aerial vehicles [Yu et al., 2011]. However, this disadvantage is acceptable in current security systems. In general, static video cameras are used with security systems based on computer vision. Some authors [Yilmaz et al., 2006, Watada et al., 2010] point out another different category of motion detectors in addition to the four already mentioned: point detectors. This

technique is primarily used to track interesting points, and is useful in stereo vision applications or for camera location in the scene, however, it is not so interesting in security environments. Background subtraction is therefore deemed the most suitable technique for this type of application.

2.2.2 Background subtraction models

Although some researchers [Zhao and Nevatia, 2004] use the simple Gaussian algorithm presented in [Wren et al., 1996], Gao et al. [Gao et al., 2000] state that a simple Gaussian is not a good model. Stauffer and Grimson [Stauffer and Grimson, 1999] presented a mixture of Gaussians model (MGM) based on the work of Wren et al.. Piccardi [Piccardi, 2004] reviews the most interesting approaches that use background subtraction models and explains that the best results, with limited resources, are attained by the MGM. Section 2.3 details how MGM technique works.

MGM has been demonstrated to be a very robust method for background subtraction. Many researchers have used it to incorporate improvements. Lee presents a new way to initialize the Gaussians to accelerate convergence to the desired value [Lee, 2005], Wu and Trivedi include a spatial correlation technique to make MGM more reliable [Wu and Trivedi, 2005], Heikkila and Pietikainen modify it using Local Binary Patterns instead of Gaussians [Heikkila and Pietikainen, 2006], Cristani et al. [Cristani et al., 2002] and Poppe et al. [Poppe et al., 2007] propose two techniques to make the MGM able to handle quick illumination changes . Tsai et al. state in [Tsai and Lai, 2009] that the MGM is used as a standard reference background subtraction algorithm by many researchers [Zhang et al., 2009, Ko, 2008, Heikkila and Pietikainen, 2006, Piccardi, 2004, Elgammal et al., 2000, Brown et al., 2005, Li et al., 2004]. Therefore, MGM has become a general background subtraction comparison model.

Nevertheless, many authors have pointed out that MGM can be outperformed in computational time and propose other background subtraction techniques. In [Kim et al., 2005], to avoid using costly operations, the authors propose modeling the background with a codebook, where a set of codewords are assigned to each pixel and new values are assigned to one of the codewords. Depending on their values, each codeword will model a foreground or background object. Zivkovic [Zivkovic and Van Der Heijden, 2004, Zivkovic, 2004] implements a way to adapt the learning rate to the number of samples, as well as a new way to calculate the weight of each Gaussian in the model. Gaussians with insignificant weight are eliminated, allowing the algorithm to reduce processing time. Yet another technique to improve computational time with respect to MGM is to reduce the detection area, by means of dynamic gates [Alvar et al., 2014a] or to segment the image into regions thanks to Mean Shift algorithms [Chen et al., 2012]. As stated in [Tsai and Lai, 2009], the updating process in background subtraction techniques may be optimized to reduce the consumed time even more, granting more time to further complex analysis [Alvar et al., 2013]. The model presented in section 2.6

proposes a new time consumption optimization, using RTDENN, detailed in section 2.4, for the updating process of background subtraction models. If no optimization is implemented, these techniques typically use low-resolution images, incurring in an unnecessary loss of interesting data.

2.2.3 Focusing on restricted areas

Cermak and Keyzer [Cermak and Keyzer, 2007] divide the images into a set of blocks. The average pixel value is calculated for each block. This value is compared with the value of the same block in the background image. If the difference is large enough, a comparison of color distribution is executed to determine if the change is due to noise or to real movement in the image. The distributions are calculated using the Mean Shift algorithm [Comaniciu et al., 2001, Comaniciu and Meer, 2002]. Piccardi [Piccardi, 2004] states that the Mean Shift algorithm is too slow and proposes some improvements [Piccardi and Jan, 2004] to make it faster. Cermak's technique focuses on a restricted area of the image to avoid wasting computational time with Mean Shift in uninteresting zones.

Liu et al. [Liu et al., 2005] introduce the concept of a virtual gate which is similar to a region of interest (ROI). Both are static, which is to say they focus on the same pixels of each image over the time. In Liu's work, virtual gates are static and allow the model to count people passing through them, ignoring the rest of the image. In the same way, Beleznai et al. [Beleznai et al., 2007] use a ROI to count passengers in an underground car. When a moving object is detected in the ROI, an optical flow is calculated only in the ROI using the past images that have been saved. The result is used to estimate the moving object's direction, so as to know if the person enters or leaves the car.

Zhao and Nevatia [Zhao and Nevatia, 2004] use "entrances and exits". These zones are the opposite of the previous ROIs or gates. Entrances and exits are scene regions where occlusions and erroneous motion detections are usual. The authors therefore propose ignoring all the moving objects detected in these regions to avoid problems.

Kim and Park [Kim and Park, 2006] presented a technique that generates a background and foreground model in each frame using the data of the "gate window" that surrounds the target. The target must be much smaller than the "gate window" analyzed to make the segmentation possible, thus requiring analysis of a large region for their proposed method to work properly. In the model presented in this thesis, the dynamic gates follow the moving object, taking as few pixels not belonging to the target as possible, to contribute even greater efficiency. In the same scene, Kim's technique would have to analyze a larger region than the model proposed in section 2.5.

2.2.4 Simultaneous motion detection and tracking

Section 2.5 proposes a time improvement model for motion detection using background subtraction; it may be used to solve motion detection and tracking simultaneously, by applying a small step of labeling the moving objects. The tracking model may also be changed easily by other more complex and robust algorithms.

Comaniciu et al. combined both steps in [Comaniciu et al., 2001] with the Mean Shift algorithm. Nonetheless, Wu and Nevatia use the Mean Shift tracker only in special cases, to avoid using such a complex algorithm regularly. These authors propose using edgelets to detect and track people [Wu and Nevatia, 2007]. Mean Shift is usually employed in cases where the edgelets cannot explain previous hypotheses of people's movements. Since the edgelets technique is a statistical learning method, it will only work with certain conditions of camera position.

Tao et al. [Tao et al., 2002] propose another technique combining tracking and detection with an expectation-maximization (EM) algorithm, called layering. The layering technique consists of a mixture of layers to model the foreground and background. Tracking is developed taking into account the shape, appearance and motion of the objects. The layering model does not work properly with articulated objects such as people, as it has been designed to track unarticulated objects like cars. This technique, although focusing tracking in zones where the object will be *a priori*, does not restrict the search to some ROIs of the image.

2.3 The Mixture of Gaussians Model

MGM is a background subtraction technique that allows the background to be modeled with more than one value for each pixel. Stauffer and Grimson [Stauffer and Grimson, 1999] [Stauffer and Grimson, 2000] propose adopting a probabilistic model for the background image. Each pixel is modeled as a mixture of weighted Gaussians. Gaussians have the same dimension M as the pixel value of the image: three in case of RGB images, and one for grayscale images. They assume that each pixel is independent of its neighboring pixels. Additionally, channels are also assumed to be independent from each other. Moreover, they consider that all the channels have the same variance. The parameters of these Gaussians are updated with the information obtained from every new frame. Finally, background segmentation is carried out based on a threshold as will be explained.

2.3.1 Update of parameters

The value of all pixels are analyzed on a frame basis to check whether it fits into one of the existing Gaussians in the pixel model. A value is considered to fit if it is between a range of λ times the standard deviation from the mean of the Gaussian. A value of 2.5 for λ is recommended [Power and Schoonees, 2002]. Each Gaussian is weighted to indicate how often new values have been considered part of it. This means that Gaussians with a higher weight will represent the values with a higher probability, i.e. values of the background. The weight is updated at each frame as follows:

$$w_{k,t} = (1-\alpha)\,\widetilde{w}_{k,t-1} + \alpha L_{k,t} \tag{2.1}$$

where $w_{k,t}$ is the weight of the kth Gaussian at time t, with $1 \leq k \leq K$, K being the total number of Gaussians that models a pixel; $\widetilde{w}_{k,t}$ is the normalized weight as shown in (2.2); α is the learning rate; and $L_{k,t}$ is a binary variable which equals 1 when the new pixel fits the kth Gaussian. Using (2.1), the weight of the Gaussian containing the new pixel is increased, while the other weights are decreased. Weights are normalized at each frame t according to:

$$\widetilde{w}_{k,t} = \frac{w_{k,t}}{\sum_{k=1}^{K} w_{k,t}} \tag{2.2}$$

$$\sum_{k=1}^{K} \widetilde{w}_{k,t} = 1 \tag{2.3}$$

The rest of the parameters needed in this model are updated just for the Gaussian k where the new pixel value fitted.

$$\begin{aligned} \boldsymbol{\mu}_{k,t} &= (1-\rho)\boldsymbol{\mu}_{k,t-1} + \rho \boldsymbol{X}_t \\ \boldsymbol{\Sigma}_{k,t} &= \sigma_{k,t}^2 \boldsymbol{I} \\ \sigma_{k,t}^2 &= (1-\rho)\sigma_{k,t-1}^2 + \rho(\boldsymbol{X}_t - \boldsymbol{\mu}_{k,t})^T(\boldsymbol{X}_t - \boldsymbol{\mu}_{k,t}) \end{aligned} \tag{2.4}$$

where $\boldsymbol{\mu}$, $\boldsymbol{\Sigma}$ and σ are the mean, covariance matrix and standard deviation of the Gaussian respectively, and $\boldsymbol{X}_t$ is the value of the pixel in frame t. As can be seen, the model simplifies the covariance matrix, assuming all the channels are independent and have the same variance.

Similarly to Equation (2.1), Equation (2.4) uses a second learning rate ρ that may be simplified as follows [Power and Schoonees, 2002]:

$$\rho = \alpha / \widetilde{w}_{k,t} \tag{2.5}$$

Initially, the value of the pixel of the first frame is taken to create the first Gaussian of the model. The mean of this new Gaussian is assumed to be the value of the pixel. When a new value does not fit any existing Gaussian, a new Gaussian is created. The model can include a maximum number of Gaussians K. If there are more than K Gaussians modeling a pixel, the new Gaussian will substitute the existing Gaussian of lowest weight. Each new Gaussian will be centered in the new value of the pixel that did not fit other Gaussians, and will have a weak weight and large standard deviation. The most significant values of the background are saved in the other Gaussians, while the new value is automatically considered foreground due to the low weight.

2.3.2 Background segmentation

The Gaussians of each pixel are sorted following the factor w/σ. This classification gives priority to the Gaussians with stronger weight and smaller deviation, that is, the ones modeling background pixels. To determine how many Gaussians code for background, a threshold T_B is used as follows:

$$B = argmin_b(\sum_{k=1}^{b} \widetilde{w}_{k,t} > T_B) \qquad 0 < T_B < 1 \tag{2.6}$$

The B first Gaussians in the ordered list are considered to be modeling background values, and the rest $K - B$ Gaussians are considered as foreground. The bigger T_B is, the more Gaussians are considered as background.

2.4 Real-Time Dynamic Ellipsoidal Neural Network

This section presents a brief theoretical description of the RTDENN model [Sánchez Miralles and Sanz Bobi, 2002, Sánchez Miralles and Sanz Bobi, 2004, Sánchez Miralles and Sanz Bobi, 2006], focusing on the training algorithm, and more specifically on the merger of two neurons, which will be essential in the update process proposed in Section 2.6.

2.4.1 RTDENN model

RTDENN is conceived as a neural network model that models an M-dimensional space R. It can be divided into subworkspaces or regions $R_k \subset R$. Let $\boldsymbol{X}_t$ be the t-th sample value of the variable monitored. Each $\boldsymbol{X}_t$ belongs to $R \subset \Re^M$.

The RTDENN consists of a set of neurons nn_k where each neuron is specialized in a particular subspace R_k. Every set of sample vectors included in every subworkspace R_k represented by neuron nn_k can be characterized by the number of samples that belong to its region n_k, the mean of the values of these samples $\boldsymbol{\mu}_k = \sum_{j=1}^{n_k} \frac{\boldsymbol{X}_k^j}{n_k}$ and the covariance matrix $\boldsymbol{\Sigma}_k$.

The RTDENN learning process is an unsupervised one step process [Specht, 1991]. This kind of learning process consists of a set of cycles where only the last elements are used to adapt the network parameters, while the rest of elements, used in previous cycles, are discarded. The learning process is thus carried out at a fast pace, as in each new cycle new samples are obtained and, once the network has been adapted, they are discarded. The weights estimation of the network maximizes the log-likelihood V_k of each neuron nn_k.

$$V_k = \frac{1}{n_k} \sum_{j=1}^{n_k} log\left(\frac{1}{\sqrt{(2\cdot\pi)^M \cdot |\boldsymbol{\Sigma}_k|} \cdot e^{-\frac{1}{2}(\boldsymbol{X}_j - \boldsymbol{\mu}_k)\boldsymbol{\Sigma}_k^{-1}(\boldsymbol{X}_j - \boldsymbol{\mu}_k)}} \right) \tag{2.7}$$

2.4.2 Update of parameters

RTDENN is suitable for real-time implementation thanks to the update process of the parameters. Every time t a new set of samples excites the network nn_k, it is updated taking into account the number N of new samples, the mean $\boldsymbol{\mu}_N$ and the covariance matrix $\boldsymbol{\Sigma}_N$ of these new values:

$$\begin{array}{l} n_{k,t} = n_{k,t-N} + N \\ \boldsymbol{\mu}_{k,t} = \frac{n_{k,t}-N}{n_{k,t}} \boldsymbol{\mu}_{k,t-N} + \frac{N}{n_{k,t}} \boldsymbol{\mu}_N \\ \sigma^2_{x,y,k,t} = \frac{n_{k,t}-N}{n_{k,t}} (\sigma^2_{x,y,k,t-N} + \mu_{x,k,t-N} \cdot \mu_{y,k,t-N}) + \\ \frac{N}{n_{k,t}} \cdot (\sigma^2_{x,y,N} + \mu_{x,N} \cdot \mu_{y,N}) - \mu_{x,k,t} \cdot \mu_{y,k,t} \end{array} \tag{2.8}$$

where $\sigma_{x,y,k,t}$ is the value of element (x, y) of $\boldsymbol{\Sigma}_{k,t}$ and $\mu_{x,k,t}$ is the value of element x of $\boldsymbol{\mu}_{k,t}$. In the original work, a forgetting factor *ff* based on [Specht, 1991] is introduced in these equations to reduce the importance of the old samples compared to the new ones, in a similar way to the learning factor α in MGM algorithm.

The excitation degree of each neuron is estimated using the inverse of the square root of the Mahalanobis distance between the mean of the sample vectors which excite

the neuron and each neuron:

$$exc_k = \sqrt{\frac{1}{(\boldsymbol{\mu}_N - \boldsymbol{\mu}_{k,t})^T \cdot (\boldsymbol{\Sigma}_{k,t})^{-1} \cdot (\boldsymbol{\mu}_N - \boldsymbol{\mu}_{k,t})}} \tag{2.9}$$

$$\begin{aligned} &max(exc_k) = exc_j \\ &exc_j > T_{exc} \end{aligned} \tag{2.10}$$

The excitation level is used to know which of the neurons should be updated: the neuron nn_j that satisfies Equation (2.10). The most excited neuron within a threshold T_{exc} is the neuron updated. If none surpass the threshold, a new neuron nn_i is added to the RTDENN. This neuron has a number of samples $n_i = N$, mean $\boldsymbol{\mu}_{i,t} = \boldsymbol{\mu}_N$ and covariance matrix $\boldsymbol{\Sigma}_{i,t} = \boldsymbol{\Sigma}_N$.

An analysis and merger of close neurons in the RTDENN is then performed. This step is an important feature because it allows the reduction of the number of neurons and thus saves memory, resources and calculation time. The fusion tries to find close neurons nn_i and nn_j using Equations (2.11) and (2.12), i.e. neurons with very similar subworkspaces.

$$d_{i,j} = \sqrt{(\boldsymbol{\mu}_{j,t} - \boldsymbol{\mu}_{i,t})^T \cdot (\boldsymbol{\Sigma}_{i,t})^{-1} \cdot (\boldsymbol{\mu}_{j,t} - \boldsymbol{\mu}_{i,t})} \tag{2.11}$$

$$d_{i,j}^2 < \chi_{M,\beta}^2 \tag{2.12}$$

The parameters of the merged neurons are calculated using Equation (2.8). If the new region obtained is too wide, the merger does not take place to avoid the region of a single neuron occupying the whole domain R. A new merged neuron is considered to be acceptable if it achieves (2.13), where v_{max} represents the minimum accuracy necessary for the prediction of the neural network

$$\frac{(\boldsymbol{\mu}_{j,t} - \boldsymbol{\mu}_{i,t})^T \cdot (\boldsymbol{\mu}_{j,t} - \boldsymbol{\mu}_{i,t})}{(\boldsymbol{\mu}_{j,t} - \boldsymbol{\mu}_{i,t})^T \cdot (\boldsymbol{\Sigma}_{i,t})^{-1} \cdot (\boldsymbol{\mu}_{j,t} - \boldsymbol{\mu}_{i,t})} < v_{max} \tag{2.13}$$

2.5 Static and dynamic gates model

This section presents a new fast model to improve MGM speed called the Static and Dynamic Gates Model (SDGM) that processes only some pixels of each image. Processed pixels are grouped in a new concept called gate. This model achieves a significant reduction in computation time, which in turn leaves more resources available for other algorithms.

In adaptive background subtraction algorithms, the updating process consumes a great amount of computational time [Tsai and Lai, 2009]. As has been already stated, it is important to reduce the computing time of the motion detection step to keep it free for more complex and resource-demanding steps such as behavior analysis. This thesis proposes a model that permits a reduction of the computing time focusing MGM on certain interesting regions of the image. Although prepared for MGM, this model is easily applicable to other background subtraction techniques. The proposed model will only make the detection and update the values in some regions, called static gates. Once a moving object is detected within a static gate, new gates will be created, following the object's movement. These gates will be called dynamic gates. This means that the proposed system will follow movements, while, at the same time, it makes the detection.

The mathematical formulation of the gates model and the methodology used to implement them, presented next, are novel contributions of this thesis. It is the first time the model has been formulated. Moreover, the time improvement obtained through its application to MGM presents a promising path to reduce time consumption in background subtraction techniques.

2.5.1 Gate properties

2.5.1.1 Definition

Given an image I_t at frame t, convex polygons are defined as $P_{it}|P_{it} \cap P_{jt} = \emptyset \ \forall i, j = 1, ..., N_P$ and $P_{it} = \delta P_{it} \cup \overset{*}{P}_{it}$ where:

$\overset{*}{P}_{it}$ is the region inside the boundary of the polygon.

δP_{it} is the boundary of the polygon.

The boundary consists of vertex $\delta P_{it} := \{p_{it}^1, ..., p_{it}^N\}$ such that $p_{it}^j \in I_t$.

A gate G_{it} is defined following $G_{it} := \left\{ p \in I_t | p \subset \overset{*}{P}_{it} \right\}$ where p are pixels.

2.5.1.2 Gate application

As stated before, MGM is used to demonstrate the advantages of SDGM. Although MGM is one of the fastest techniques, computing K Gaussians in each pixel of each frame of a video sequence is a time consuming task. The combination of MGM with SDGM achieves a reduction of computing time, because MGM is only applied over the gates.

Gates can be defined for any convex polygon shape, but it is recommended to choose one that is easy to process, for example, a rectangle.

2.5.2 Static gates

2.5.2.1 Description of static gates

Static gates (SG) are image regions from which moving objects enter the scene. They stand for entrances and exits of the scene. Every image of a scene has the same static gates: $SG_{it} := SG_{i(t+1)}$. If there is no movement in the image, only pixels of the SG are processed. Therefore, the speed performance of background subtraction algorithms is drastically improved.

It should be pointed out that it will only be worth using SDGM if the static gates correspond to a small part of the whole image, and they are placed in all the regions where movement can start, such as doors, windows and some of the borders of the image (see Figure 2.1). If static gates are bigger than needed, computation time will be unnecessarily increased, whereas making them too small will result in poor detection results.

Figure 2.1. Example of static gates (left) and dynamic gates (right).

2.5.2.2 Initialization

The initialization of static gates depends on the application. For example, in an outdoor security system for a house, it is interesting to detect cars entering through the garage door, but not planes flying in the sky.

There are two types of initialization: automatic and manual initialization. In manual initialization, the user has to locate the static gates previously. This is recommended when not every region where a movement could start is of interest, as stated before, or when there are interesting zones where movement does not usually start, such as a window. The selective location of static gates achieves focus of interest on particular regions. Automatic initialization consists of running the motion detection algorithm without using static gates. Every region where a movement has started becomes a static gate.

2.5.3 Dynamic gates

2.5.3.1 Definition for any polygon

A dynamic gate (DG) is a gate that, contrary to a static gate, is not persistent. Once it is created, it can change position and size from frame to frame, until it is eliminated.

Dynamic gates follow three rules:

- $DG_{it} :\neq DG_{(i+1)t}$, and $DG_{it} :\neq DG_{i(t+1)}$ in general.
- At least H pixels p of DG_{it} change their value due to a movement.
- They are created in a static gate or in a dynamic gate.

Dynamic gate updating process consists of the following steps:

1. Estimation of a base polygon for the following dynamic gate.

$$\delta P_{it} = H_{convex}(SA(DG_{it})) \tag{2.14}$$

where:

$SA(DG_{it})$ is the resulting blob after the execution of the subtraction algorithm on the gate.

H_{convex} is the convex hull calculated using the pixels of the blob.

δP_{it} is the boundary of the base polygon.

2. Estimation of the movement vector of the polygon centroid:

$$\mathbf{v} = centroid(\delta P_{it}) - centroid\left(\delta P_{i(t-1)}\right) \tag{2.15}$$

3. Estimation of the boundary of the polygon of the new dynamic gate:

$$\delta \hat{P}_{i(t+1)} = H_{convex}\left(\{p^1_{it}, ..., p^N_{it}, p^1_{it} + \mathbf{v}, ..., p^N_{it} + \mathbf{v}\}\right) \tag{2.16}$$

where:

$\hat{P}_{i(t+1)}$ is the polygon corresponding to the estimated position of moving object i in frame $t+1$. This estimation is the dynamic gate, where MGM will be carried out. $\hat{P}_{i(t+1)}$ is always bigger than $P_{i(t+1)}$. While $\hat{P}_{i(t+1)}$ is where the object can be located, $P_{i(t+1)}$ is where the object really is located (see Figure 2.2).

4. If $\exists i, j, i \neq j | \hat{P}_{i(t+1)} \cap \hat{P}_{j(t+1)} \neq \emptyset$, then:

$$\delta \hat{P}_{i(t+1)} = \hat{P}_{j(t+1)} = H_{convex}\left(\left\{p^1_{it}, ..., p^1_{it} + \mathbf{v}, ..., p^1_{jt}, ..., p^1_{jt} + \mathbf{v}, ...\right\}\right) \tag{2.17}$$

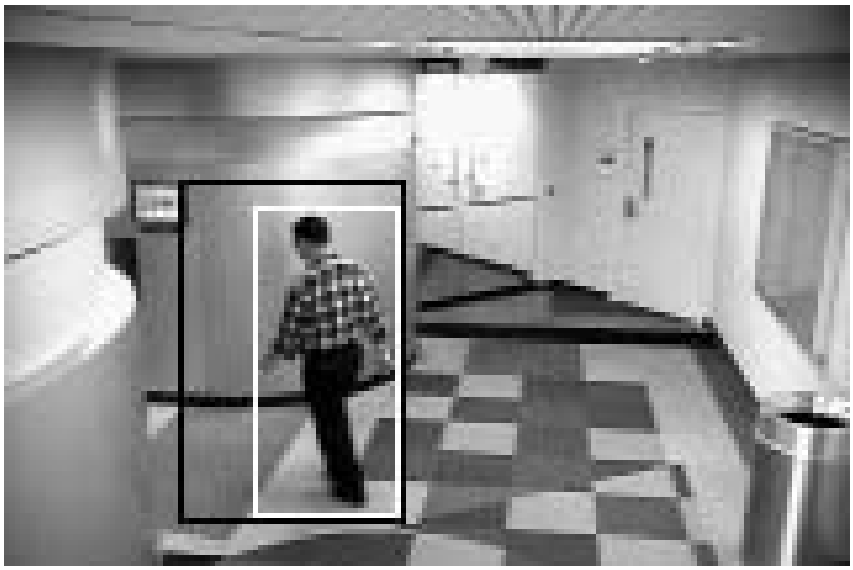


Figure 2.2. Example of P_{it} (white) and $\hat{P}_{i(t+1)}$ (black).

It should be pointed out that $SA(DG_{it})$ could result in several blobs and thus these steps should be executed several times, creating new dynamic gates.

2.5.3.2 Particular simple definition for a rectangle

The general mathematical formulation of the gates for any kind of polygon has been shown, however it is usual and less resource-demanding to use the simplest polygon possible. If a rectangle is chosen as the boundary of the polygons, the previous steps can been simplified as follows:

1. Estimation of a rectangle for the following dynamic gate.

$$\delta P_{it} = Encrec\,(SA\,(DG_{it})) \tag{2.18}$$

   where:

   $SA\,(DG_{it})$ is the resulting blob after the execution of the subtraction algorithm on the gate.

   $Encrec$ is the enclosing rectangle of the blob.

   δP_{it} is the boundary of the rectangle.

2. Estimation of the movement vector of the rectangle centroid:

$$\mathbf{v} = centroid\,(\delta P_{it}) - centroid\left(\delta P_{i(t-1)}\right) \tag{2.19}$$

3. Estimation of the boundary of the rectangle applying an affine transformation:

$$\delta \hat{P}_{i(t+1)} = Encrec\left(\{p_{it}^1, ..., p_{it}^N, p_{it}^1 + \mathbf{v}, ..., p_{it}^N + \mathbf{v}\}\right) \tag{2.20}$$

2.5.3.3 Description of dynamic gates

Moving objects are firstly detected within static gates. They contain blobs of moving objects, obtained by executing the MGM. Dynamic gates are then created to follow the moving objects throughout the video sequence (see Figure 2.1). Due to a crossing of moving objects, two dynamic gates may be merged, or inversely, if a group of people divides, new dynamic gates may be created. While static gates remain throughout the video sequence, dynamic gates are created when a new object is detected in a static gate, or when a moving object splits into several objects in another dynamic gate. The algorithm calculates the object speed taking into account the present and previous frames. This generates an estimation of the next location of the object ($\hat{P}$), where the dynamic gate will be placed[1]. The MGM is applied to the pixels belonging to a dynamic gate, as in the case of static gates.

2.5.4 Pixels in a gate

The pixels belonging to a gate are the only pixels that are candidates to be considered as foreground. In the same way, they are the only pixels that will be updated. Tsai et al. explain that the update step consumes a great amount of time, while not using adaptive methods allows the detection of people who do not move for a long period of time, such as an unconscious person or people watching TV [Tsai and Lai, 2009].

There are two simple improvements that can be made in SDGM to solve this problem without losing system adaptability. Firstly, as with other background subtraction techniques, parameter updating can be executed only in background zones. This way, the background is updated without introducing foreground elements. Elgammal et al. present a problem using that solution [Elgammal et al., 2000]. Secondly, the SDGM may be forced to keep the dynamic gates active until the moving object has reached a static gate. This means that the dynamic gate will persist in the scene as long as the person does not exit the scene using one of the static gates defined. In that way people are detected even though they stand still for long periods of time.

2.5.5 Other Pixels

The pixels not belonging to any gate are defined with:

[1] see http://www.iit.upcomillas.es/malvar/SDGM.php

$$p \in I_t | p \not\subset \overset{*}{P}_{it}, \forall i \tag{2.21}$$

No motion detection is carried out on those pixels. Nevertheless, they should be updated since they are part of the background, which can change due to many causes, such as changes in illumination, falling leaves, etc. Updating those pixels should not be performed in every frame, in order to avoid wasting too much computing time.

The pixels not belonging to a gate are updated every F frames. F is the only new parameter that SDGM introduces to MGM. Equation (2.22) is similar to Equation (2.4), however, it uses the parameters α_{NG} and ρ_{NG}, corresponding to α and ρ, but for pixels not belonging to a gate. The value of these parameters are:

$$\begin{aligned} \alpha_{NG} &= F * \alpha \\ \rho_{NG} &= \alpha_{NG} / \widetilde{w}_{k,t} \end{aligned} \tag{2.22}$$

The weights are updated as follows,

$$w_{k,t} = (1 - \alpha_{NG})\, \widetilde{w}_{k,t-1} + \alpha_{NG} L_{k,t} \tag{2.23}$$

and the parameters of the active Gaussian:

$$\begin{aligned} \mu_t &= (1 - \rho_{NG})\mu_{t-1} + \rho_{NG} x_t \\ \sigma_t^2 &= (1 - \rho_{NG})\sigma_{t-1}^2 + \rho_{NG}(x_t - \mu_t)^2 \end{aligned} \tag{2.24}$$

Every F frames, but also the new pixel value must be fitted to the Gaussians; in addition to the creation or deletion of Gaussians if necessary. As mentioned, motion detection is the only step that will not be performed in these pixels, because they will be considered to be background as long as a moving object does not approach them.

2.6 Mixture of merged Gaussians algorithm

This section sets forth an adaptive approach for a segmentation algorithm based on the probabilistic background model of MGM and the learning process of the RTDENN for the update of the model.

This approach models the background scene through a sum of Gaussians, which allows the model to deal with different backgrounds in differetn conditions. This means that cyclical changes will be considered to be part of the background. For instance, in the case of swinging tree leaves, the background model would consider both the color of the leaves and the color of the sky as background. Furthermore, the background

model is updated so that it can adapt itself to permanent changes in the scene. Taking the previous example, the model would take darker colors of green and blue for the background as the night falls.

The background model is composed of a mixture of Gaussians to represent each pixel, similarly to MGM. Segmentation is performed at every frame by comparing each pixel value to its corresponding background model in order to determine whether the pixel belongs to the background or, on the contrary, to the foreground. Nevertheless, the update of the model is performed adopting a similar solution to the RTDENN training algorithm. This novel approach applied to the updating process of a background subtraction model is a contribution of this thesis. The RTDENN differs from the update of the background model used by MGM, permitting a faster updater background model update. The main advantage of the MMGA is the reduction of computing time without losing accuracy with respect to MGM, the most important objective of the presented background segmentation algorithms.

The following subsections further explain the MMGA, and focus on the main and innovative aspects of the algorithm, namely the creation and update of the background model and background/foreground segmentation.

2.6.1 Background model

A pixel is represented by its value $\boldsymbol{X}_t$ at the instant t, which is a vector with the M components given by the channels of the input video (e.g.: one component for gray scale videos, or three different components for RGB videos), as expressed in Equation (2.25). $X_{m,t}$ is the value of channel m of vector $\boldsymbol{X}_t$.

$$\boldsymbol{X}_t = \{X_{1,t}, X_{2,t}, ..., X_{M,t}\} \tag{2.25}$$

The temporal series of values $\boldsymbol{X}_t$ of a pixel over a certain period of time is considered a stochastic process, and is stored by the algorithm to build a sample of size N. Thus, a new Gaussian g_i is created with 3 parameters: number of observations ($n_{i,N}$), mean ($\boldsymbol{\mu}_{i,N}$) and covariance ($\boldsymbol{\Sigma}_{i,N}$). The number of observations associated to this Gaussian is $n_{i,N} = N$. The mean $\boldsymbol{\mu}_{i,N}$ and covariance $\boldsymbol{\Sigma}_{i,N}$ are estimated by the unbiased estimators sample mean and sample covariance given by Equations (2.26), (2.27) and (2.28) for the fitting of the sample data in a Gaussian distribution. Input video channels are considered to be independent, so that the covariance matrices for the Gaussians are diagonal matrices with the variance of each input channel in the main diagonal, as expressed in (2.27). This assumption leads to a remarkable reduction of the computational cost with no significant effect on the accuracy and reliability of the segmentation results, as explained in [Stauffer and Grimson, 1999]. However, contrary

to the MGM method, the proposed method permits different variance for each channel, thus allowing the use of different color spaces such as HSV (used in [Cucchiara et al., 2003]), or color and brightness (used in [Horprasert et al., 1999, Kim et al., 2005]).

$$\boldsymbol{\mu}_{i,N} = \frac{\sum_{t=1}^{N} (\boldsymbol{X}_t)}{N} \tag{2.26}$$

$$\Sigma_{i,N}(x,y) = \begin{cases} \sigma^2_{x,i,N} & \forall x = y \\ 0 & \forall x \neq y \end{cases} \tag{2.27}$$

$$\sigma^2_{x,i,N} = \frac{\sum_{t=1}^{N} (X_{t,x} - \mu_{x,i,N})^2}{N-1} \tag{2.28}$$

The process of data storing is successively repeated, so that every N frames, a new Gaussian g_i is created for each pixel. The background model is updated to take into account the information provided by the new Gaussian g_i. In order to accomplish the data accommodation, this new Gaussian g_i is compared to the current mixture of Gaussians that comprise the background model at that instant. This comparison will result in either merging the new Gaussian with an existing one, or directly incorporating the new Gaussian into the model if the required conditions for the merger are not satisfied. The model may be composed of a maximum number of Gaussians K, so that when the model is already composed by K Gaussians and a new Gaussian must be included, the latter will substitute the Gaussian that has the least number of observations associated, which is equivalent to the weight in MGM. But, contrarily to MGM, the background model is updated with a new Gaussian once every N frames, instead of considering a new value pixel at every frame.

The update of the background model is based on the learning algorithms of RTDENN, where the neurons comprising a network excited by a set of samples are updated depending on the excitation level, which is the inverse of the square root of the Mahalanobis distance between the mean of the sample vectors which excite the neuron and each neuron. If the excitation level surpasses a certain threshold, the new data is merged into the neuron; otherwise, a new neuron with the new data is created and added to the RTDENN. The merger of close neurons is an important feature because it allows a reduction in the number of neurons and thus saves memory, resources and calculation time.

In this model, the merger of the new Gaussian g_i created with the data of the last N observations and the Gaussian g_j that was already part of the background model will be developed only under the two following conditions:

1. The merging Gaussians g_i and g_j must comprise very close regions.

2. The region of the resulting Gaussian g_h must be small enough.

The proximity of two Gaussians is measured by the Mahalanobis distance, as in Equation (2.11), which defines the distance from Gaussian g_i and Gaussian g_j as stated in Equation (2.29):

$$d_{i,j} = min\left(\sqrt{(\boldsymbol{\mu}_{j,t}-\boldsymbol{\mu}_{i,N})^T(\boldsymbol{\Sigma}_{i,N})^{-1}(\boldsymbol{\mu}_{j,t}-\boldsymbol{\mu}_{i,N})}, \sqrt{(\boldsymbol{\mu}_{j,t}-\boldsymbol{\mu}_{i,N})^T(\boldsymbol{\Sigma}_{j,t})^{-1}(\boldsymbol{\mu}_{j,t}-\boldsymbol{\mu}_{i,N})}\right) \tag{2.29}$$

The first condition of proximity is checked using the square of the Mahalanobis distance, $d^2_{i,j}$, which has a Chi-Square distribution. The hypothesis test given by Equation (2.30) is carried out, where M is the number of degrees of freedom, which corresponds to the number of input channels, and β is the significance level of the test, usually taken as 95%. If the result of the test is positive, the parameters of the merged Gaussian g_h are calculated with Equations (2.31), (2.32) and (2.33):

$$d^2_{i,j} < \chi^2_{M,\beta} \tag{2.30}$$

$$n_{h,t} = (1-\alpha)n_{i,N} + \alpha n_{j,t} \tag{2.31}$$

$$\boldsymbol{\mu}_{h,t} = \frac{n_{i,N}}{n_{i,N}+n_{j,t}} \cdot \boldsymbol{\mu}_{i,N} + \frac{n_{j,t}}{n_{i,N}+n_{j,t}} \cdot \boldsymbol{\mu}_{j,t} \tag{2.32}$$

$$\sigma^2_{x,h,t} = \frac{n_{i,N}}{n_{i,N}+n_{j,t}} \cdot (\sigma^2_{x,i,N} + \mu^2_{x,i,N}) + \frac{n_{j,t}}{n_{i,N}+n_{j,t}} \cdot (\sigma^2_{x,j,t} + \mu^2_{x,j,t}) - \mu^2_{x,h,t} \tag{2.33}$$

These equations are similar to the merger equations of RTDENN, but they include the learning rate of MGM α.

The second condition to ensure the precision required by the system is related to the dispersion of the data provided by the observations that constitute the Gaussian. To evaluate this condition, the variances of the new Gaussian are compared to a maximum set by the parameter v_{max}, as stated in Equation (2.34).

$$\frac{(\boldsymbol{\mu}_a-\boldsymbol{\mu}_{h,t})^T \cdot (\boldsymbol{\mu}_a-\boldsymbol{\mu}_{h,t})}{(\boldsymbol{\mu}_a-\boldsymbol{\mu}_{h,t})^T \cdot (\boldsymbol{\Sigma}_{h,t})^{-1} \cdot (\boldsymbol{\mu}_a-\boldsymbol{\mu}_{h,t})} < v_{max} \qquad \forall a = \big\{\{i,N\},\{j,t\}\big\} \tag{2.34}$$

If both Gaussians g_i and g_j also satisfy this condition, the merger is completed, substituting the existing Gaussian g_j by the resulting Gaussian g_h in the background model.

ν_{max} and N, as explained in [Sánchez Miralles and Sanz Bobi, 2002], are the main parameters of the training process. The former deals with precision, while the latter is concerned with execution time. On one hand, ν_{max} is a limit of noise acceptance. On the other hand, high values of N are expected to reduce execution time, to the detriment of segmentation accuracy.

Although theoretically ν_{max} is important to avoid Gaussians with excessively high standard deviations, it can be seen experimentally that the value does not significantly modify the results. For a scale of value of 0 to 255, ν_{max} has been set to 30.

Another parameter is introduced into the model to solve some problems in very static backgrounds and with low noise cameras, where the standard deviation becomes too low. The parameter ν_{min} is the lower limit for covariance of the Gaussian distribution that comprises the model. Since input video channels have been considered to be independent from each other, covariance is composed only by the standard deviation for each channel, limited by ν_{min}. The use of minimum values for covariance has already been explained in other works such as [Horprasert et al., 1999]. A low value of ν_{min}, makes the model very precise, but without any ability to generalize, while high values of ν_{min} make the model very imprecise.

2.6.2 Background segmentation

A pixel will be part of the background if its value is contained in the most probable scene given by the background model, as is proposed in MGM [Stauffer and Grimson, 1999]. The most probable scene is a set B of Gaussians that models the background. In order to determine B, the Gaussians are sorted in decreasing order of relevance. Relevance in the background model is given by parameter C_k, which considers the relative number of observations associated to a Gaussian g_i and its standard deviation, as stated in Equation (2.35). This set of B Gaussians includes at least a percentage T_B of the total data accounted for by the neural network, as can be seen in Equation (2.36).

$$C_k = \frac{n_{k,t}}{|\Sigma_{k,t}|} \tag{2.35}$$

$$B = argmin_b \left(\frac{\sum_{p=1}^{b} n_{p,t}}{\sum_{k=1}^{K} n_{k,t}} > T_B \right), \quad 0 < T_B < 1 \tag{2.36}$$

Low values of threshold T_B would lead to a unimodal most probable background scene, i.e. composed of a single Gaussian, which would be unable to deal with repetitive, irrelevant motion in the background. In contrast, very high values of T_B would result in a most probable scene comprising multiple distributions, which would cause a

transparency effect.

The pixel under consideration is part of the background if there is a connection with one of the Gaussians in the set of B first Gaussians. This connection is considered in terms of distance to the mean of the Gaussian, which must not exceed a maximum value proportional to the standard deviation of the Gaussian, with proportionality constant as stated in Equation (2.37).

$$|X_{x,t} - \mu_{x,i,t}| \leq \lambda \sigma_{x,i,t} \quad ; \quad \forall\, x \,/\, 0 \leq x \leq M,\, g_i \in B \tag{2.37}$$

The parameter λ determines the maximum distance of a pixel value to the mean of the Gaussian distribution so that the pixel is considered part of the background. The value assigned to this parameter may have a significant impact on the resulting segmentation, since it is the key for the categorization of pixels. The optimal value depends on several factors, such as the degree of disturbance in the input video, the lighting, the colors in the scene, etc. Higher degrees of noise will require higher values of λ so that small changes in the background are not considered to be relevant motion. Darker areas usually have a lower degree of variability than lighter zones.

If no connection is detected to any of the Gaussians in B, the pixel analyzed is not a part of the background, so the corresponding output will be set to foreground.

2.7 Experiments and results

This section describes some of the experiments that have been carried out to analyze the speed performance of both SDGM and MMGA, and their comparison to standard MGM. Although it is not the aim of these methods, a robustness comparison centered on time optimization was also performed. Experiments were run on an Intel Core 2 Quad 3 GHz CPU, 3.5 GB RAM and Windows XP. All the videos used in the experiment can be found on the IBM PeopleVision research team webpage [Hampapur et al., 2003]; some of the videos were published for the Second IEEE International Workshop on Performance Evaluation of Tracking and Surveillance (Kauai, December 2001). The selected set of test videos comprises different characteristics: indoor and outdoor scenes, several frames per second (fps) rates and diverse resolution values ($160 * 120$, $320 * 240$). Table 2.1 presents the four videos used for the experiments presented in this section. Additionally, the ground truth of the foreground of these videos was manually created to serve as a reference for the evaluation of the segmentation results. Accuracy of the segmentation was assessed comparing the resulting segmentation yielded by the segmentation algorithms to the ground truth. The ground truth was manually created for the test videos.

Table 2.1. Videos used.

Video	Name	fps	time	resolution
Video 1	"TwoPerson_Line_Circle_Comp_0_Quad0.avi"	30	13s	320 x 240
Video 2	"PetsD2TeC2.avi"	30	94s	320 x 240
Video 3	"IndoorGTTest2.avi"	30	116s	320 x 240
Video 4	"ThreePerson_Circles_Comp_0_Quad0.avi"	39	13s	160 x 120

First, the algorithms were implemented in Matlab, using Simulink and the Video and Image Processing Toolbox to confirm the models were robust and fast enough compared to MGM. Then, the models, including MGM, were implemented in C++, using OpenCV in order to obtain the time and error analysis results here presented. Time and error rate values for each of the three models were calculated from the C++ version of the software developed, executed in the previously described machine. Time performance calculatioin was carried out using the Windows system function `QueryPerformanceCounter()`. Unlike `clock()` or `GetTickCount()` functions, which have at most a precision of 16ms, `QueryPerformanceCounter()` have a precision under microseconds[2]. This function returns a measurement of activities that occur during a time interval that is defined by a start (motion detection in a frame begins) and an end event (motion detection ends in that frame). Time calculation did not take into consideration the time spent acquiring or decompresing the image and was performed using a single thread. Time and error rate values correspond to the mean values of fifteen executions for each of the three models: MGM, SDGM and MMGA. The ratios vwere calculated comparing the presented models (SDGM and MMGA) to MGM results. The ground truth images, codes and results developed or created during the thesis are available on the author's webpage[3].

2.7.1 SDGM results

2.7.1.1 Speed performance

Table 2.2 shows different values for the most relevant parameters: T_B being the threshold to decide how many Gaussians model the background (see equation 2.6) and F being the update rate in SDGM (see subsection 2.5.5). Cases $C1$ to $C4$, $C6$ to $C9$ and $C10$ to $C13$ correspond to the cases in the 3 videos with F varying from 25 to 200. Case $C5$ is introduced to show that SDGM decreases time independently of MGM parameters, such as T_B. Case $C14$ shows that time reduction is also independent of the size of the video input.

Performance is illustrated in Table 2.3. The first column shows the case name.

[2] http://msdn.microsoft.com/en-us/library/windows/desktop/ms644904%28v=vs.85%29.aspx
[3] http://www.iit.upcomillas.es/malvar/

Table 2.2. Parameters value for each case.

Case	Video	size	F	T_B
C1	Video 1	160*120	25	0.5
C2	Video 1	160*120	50	0.5
C3	Video 1	160*120	100	0.5
C4	Video 1	160*120	200	0.5
C5	Video 1	160*120	25	0.7
C6	Video 2	320*240	25	0.7
C7	Video 2	320*240	50	0.7
C8	Video 2	320*240	100	0.7
C9	Video 2	320*240	200	0.7
C10	Video 3	160*120	25	0.7
C11	Video 3	160*120	50	0.7
C12	Video 3	160*120	100	0.7
C13	Video 3	160*120	200	0.7
C14	Video 3	320*240	25	0.7

Columns 'SDGM' and 'MGM' display the computational time taken by each method. The 'Time ratio' column corresponds to the time used by SDGM compared to MGM, where time consumed by SDGM is divided by the time used in MGM. The 'Analyzed points' column is the ratio: the mean of points analyzed by SDGM divided by the total points of each frame. The 'SG' column shows the percentage of points analyzed when there is no movement. This number represents the area of the image occupied by the static gates. The viability of SDGM is only interesting if the error ratio does not increase. Subsection 2.7.1.2 provides verification of the constancy of the error ratio.

Observing the Speed ratio and Analyzed points columns, it can be concluded that time reduction is approximately proportional to the mean number of pixels analyzed during the video sequence. It is worth using SDGM in sparsely crowded scenes, where there are fewer moving pixels than static ones. It is important to note that in every case, the execution time is reduced considerably.

2.7.1.2 Robustness

A comparison between results obtained with MGM and SDGM shows that improvement in speed does not carry an increase in error rate. This comparison not only focuses on calculating the total number of errors of each model, but also on the number of wrongly classified pixels that SDGM solves or introduces. This is called "extra error". Extra error is calculated as errors made by one model, and not made by the other model.

Figure 2.3 shows an example of extra error. The black circle is the ground truth real moving object that the model should detect. Assuming the white circle with dotted black border is what MGM detects and the white circle with solid black line is what

Table 2.3. Time consumed for each case in both methods.

Case	SDGM [s]	MGM [s]	Time ratio [%]	Analyzed points [%]	SG [%]
C1	12.71	37.4	34.0	33.8	15.5
C2	12.27	37.4	32.9	32.5	15.5
C3	12.24	37.4	32.7	32.1	15.5
C4	12.18	37.4	32.6	32.4	15.5
C5	12.68	37.17	34.1	33.3	15.5
C6	460.1	1164.2	39.5	38.6	13.6
C7	459.9	1164.2	39.5	38.2	13.6
C8	473.7	1164.2	40.7	39.3	13.6
C9	481.5	1164.2	41.4	40.3	13.6
C10	56.1	158.5	35.4	35.4	25.0
C11	55.3	158.5	34.9	34.3	25.0
C12	53.6	158.5	33.8	33.7	25.0
C13	54.5	158.5	34.4	33.9	25.0
C14	211.9	625.9	33.9	32.4	25.0

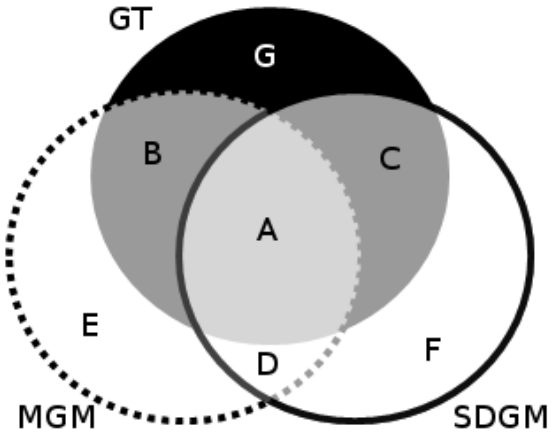


Figure 2.3. Extra error example.

SDGM detects, seven regions can be defined:

- Zones A and C are correct detection by SDGM, while D and F are incorrect ones. Zones B and G are zones not detected by SDGM as a moving object that should have been detected (false-negatives).
- In the same way, zones A and B are correct, and D and E incorrect, detections by MGM. Zones G and C are its false-negatives.
- Zone D represents the common false-positives, while zone G corresponds to false-negatives made by both methods.

- Zones C and E are the extra error made by MGM and solved in SDGM.
- Zones B and F are the extra error introduced by SDGM compared to MGM.

The sum of MGM and SDGM extra error in a frame is the total number of pixels considered differently by each model, i.e. pixels that one model considers as background and the other as foreground. If both models classify a pixel wrongly, "extra error" (zones D and G) will not increase. Extra error not only gives information about the decrease or increase in the error made by the new model, but also shows additional interesting information. It is possible to check if the new model introduces many new errors that are not made in the previous model or if it just solves some errors.

Table 2.4. Error made by SDGM and MGM for different values of F.

	Error MGM [%]	Error SDGM [%]			
F		25	50	100	200
V. 1	1.44	1.42	1.43	1.46	1.64
V. 2	1.87	0.7	0.76	0.92	1.02
V. 3	0.64	0.61	0.62	0.63	0.66

Table 2.5. Extra error made by SDGM and MGM for different values of F.

	Extra error MGM [%]				Extra error SDGM [%]			
F	25	50	100	200	25	50	100	200
V. 1	0.034	0.035	0.028	0.049	0.008	0.026	0.044	0.245
V. 2	1.202	1.191	1.135	1.089	0.031	0.075	0.183	0.23
V. 3	0.039	0.038	0.04	0.044	0.01	0.018	0.031	0.068

Table 2.5 shows the mean extra error per frame for different videos and values of F. These errors are calculated before any postprocessing algorithm is applied. Figures 2.4 and 2.5 show extra error for the two different cases. The first figure illustrates how, in the initial frames where there is no movement, only the MGM has extra error. SDGM only focuses on the static gates, thus avoiding the introduction of false-positives (pixels considered as foreground when they are not) in the image. The biggest difference between the two models is observed in frame 900 (see Figure 2.6). In this figure, as in Figure 2.7, left images correspond to real frame, upper right to SDGM and bottom right to MGM.

MGM detects the reflection of a person on the shiny wall surface as a moving object. This surface is not part of SDGM's gates, so the false-positives are not detected by this model (see Figure 2.6).

Figure 2.5 shows the performance for video 1, the poorest performance of SDGM in all the presented examples. As in the previous figure, there are fewer errors in the first frames using SDGM. Afterward, the error in SDGM increases considerably. Due to

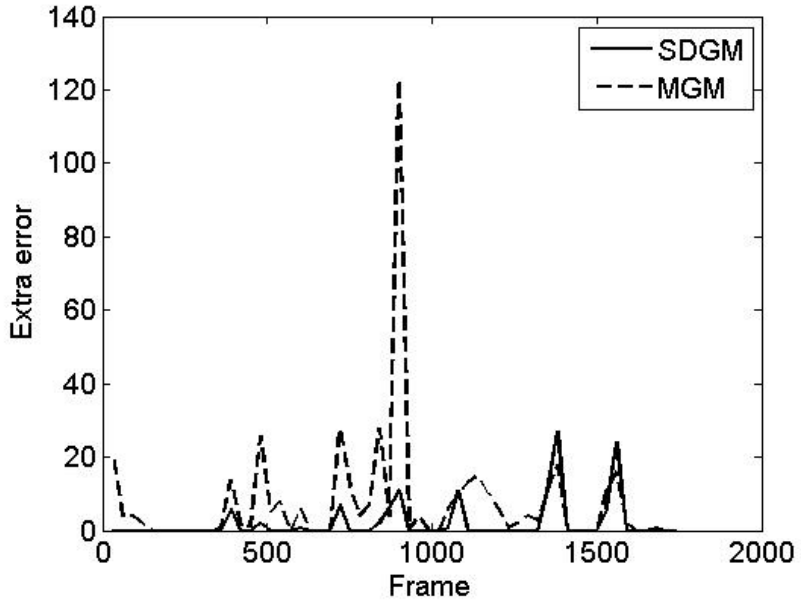


Figure 2.4. Extra error per frame in video 3 with $F = 25$.

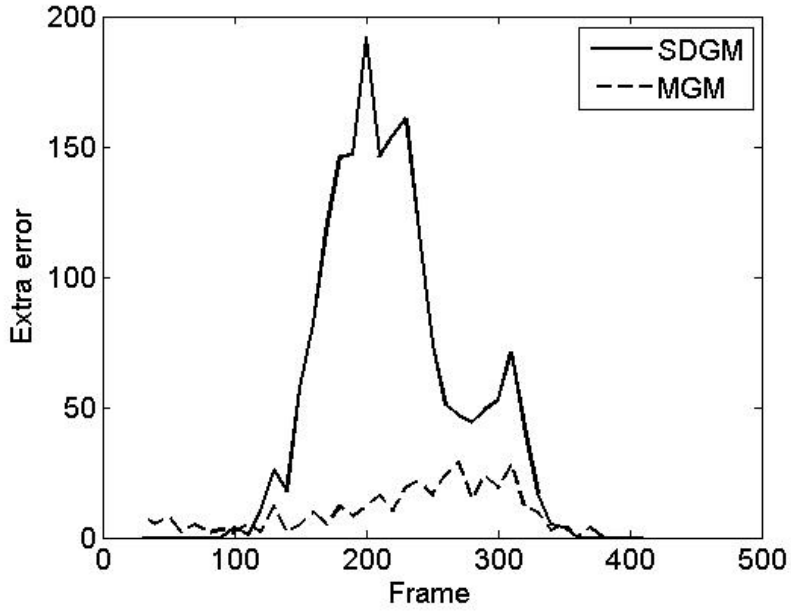


Figure 2.5. Extra error per frame in video 1 with $F = 200$.

the lack of updating, the background Gaussian does not have a short enough standard deviation, and the model considers pixels belonging to a person as background (see Figure 2.7).

The main parameter of SDGM is F, number of frames without updating for the pixels not in gates, see (2.21). The higher the value of F is, the bigger the error. The value of F must be small enough to avoid increasing the error produced by the presented model. Any value of F lower than the frames corresponding to 3 seconds is deemed acceptable (see Table 2.4).

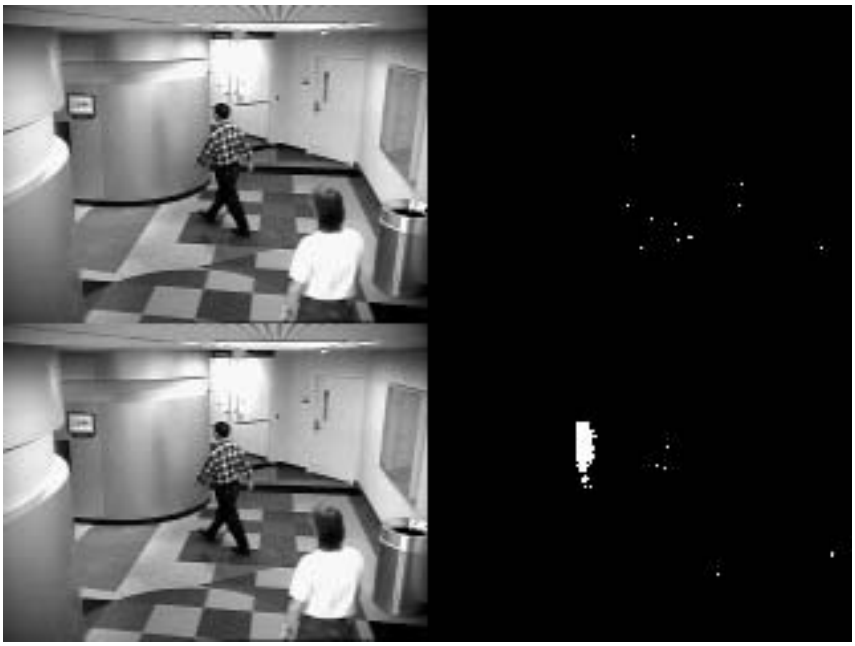


Figure 2.6. Frame 900 of video 3 and extra errors in SDGM (top) and MGM (bottom).

Figure 2.7. Frame 200 of video 1 and extra errors with $F = 200$ in SDGM (top) and MGM (bottom).

As can be seen, there is not a large difference for values of F lower than 100. That means that the updating of the whole image carried out using Equations (2.23) and (2.24) must be made once every 50 or 100 frames, which is equivalent to performing an update approximately every 2 or 3 seconds. Figure 2.8 shows the example of video 2 using F=25, compared with the errors made by MGM. The solid black line shows SDGM errors and the dashed black line MGM errors. Both lines have peaks at the same points, but SDGM produces less error at each frame.

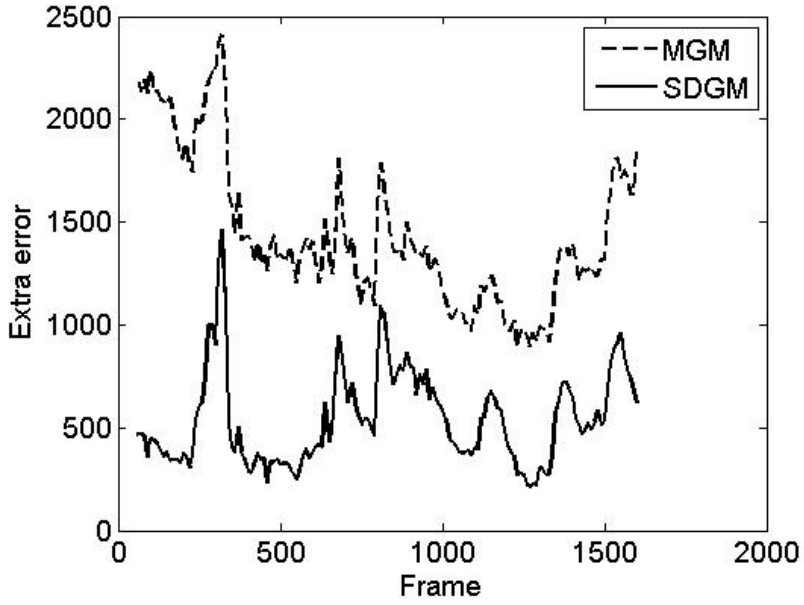


Figure 2.8. Error per frame in video 2.

2.7.2 MMGA results

First, in order to show the results depending on the numerical values for the parameters in the proposed algorithm, a set of tests was developed and a sensitivity analysis of the results for these parameters was performed. Moreover, a manually made ground truth was prepared. Then, the performance of the proposed algorithm was evaluated in terms of execution speed, reliability, and accuracy of the resulting segmentation. To facilitate this evaluation, the MGM algorithm was also implemented, and a comparison between both algorithms was established. The proposed algorithm and MGM were implemented in Matlab, using Simulink and the Video and Image Processing Toolbox.

As the segmentation algorithm proposed is the same as the one in Stauffer's MGM, the parameters λ, T_B, K and α are the same for both algorithms. These parameters have the same values as those used previously, which were the parameters proposed in Stauffer and Grimson [Stauffer and Grimson, 1999], and Power and Schoonees [Power and Schoonees, 2002]: $\lambda = 2.5$, $T_B = 0.7$, $K = 3$ and $\alpha = 0.005$. Moreover, the color space used is RGB, as was used in Stauffer's work.

2.7.2.1 Parameter sensitivity

A set of tests was carried out to analyze the effect of the parameters on the results. In order to set reasonable values for the parameters, each of them was independently analyzed, that is, tests were conducted using several different videos, changing the value of one parameter at a time, considering all other parameters as a fixed value until the best results were achieved.

Channel accuracy (ν_{min})

Several tests were carried out to estimate the proper value of ν_{min}. As an example, Figure 2.9 presents an image of the resulting segmentation for video 2 with values of $\nu_{min} = 1.5$, $\nu_{min} = 5$ and $\nu_{min} = 10$. The first choice results in a very complete segmentation, but one which is not robust against changes in lighting, as the selected image presented in Figure 2.9 shows. In contrast, for $\nu_{min} = 10$ the effect of lighting changes is suppressed, but the figure in motion is vague and incomplete. An intermediate value for ν_{min}, such as $\nu_{min} = 5$, seems to be reasonable, providing a complete segmentation and few false-positives in the form of isolated pixels, which may be easily ignored or removed in a post-processing phase.

Figure 2.9. Tests for different values of ν_{min} on frame 310 in video 2. Left to right: (i) $\nu_{min} = 1.5$, (ii) $\nu_{min} = 5$ and (iii) $\nu_{min} = 10$.

Figure 2.10 shows an image of the resulting segmentation for video 1 for $\nu_{min} = 5$, $\nu_{min} = 8$ and $\nu_{min} = 10$. For values from $\nu_{min} = 5$ the false-positive rate is reasonably low, but as the value of this parameter increases, the segmentation becomes more and more incomplete.

Figure 2.10. Tests for different values of ν_{min} on frame 250 in video 1. Left to right: (i) $\nu_{min} = 5$, (ii) $\nu_{min} = 8$ and (iii) $\nu_{min} = 10$.

Finally, this parameter has been set to the maximum critical value that enables noise to be considered as part of the background by the algorithm in order to achieve the highest degree accuracy possible, selecting $\nu_{min} = 5$.

Number of frames for the update of the background model (N)

First, a wide range of values was tested for video 4, to serve as a reference for a qualitative approach. Table 2.6 and Figure 2.11 show the mean values of the test carried out for each value. The first line in Table 2.6 corresponds to the results obtained with the MGM algorithm, added for purposes of comparison. Note that although MMGA may seem to work extremely badly with $N = 50$ in Figure 2.11, this is merely a one-frame

effect that occurs during the update of the model and is mitigated over time, as can be observed in the results presented in Table 2.7.

Table 2.6. Mean time consumed with different values of N compared to MGM (I)

N	Execution time
1 (MGM)	100%
10	38%
20	19%
30	15%
40	12%
50	11%
60	11%

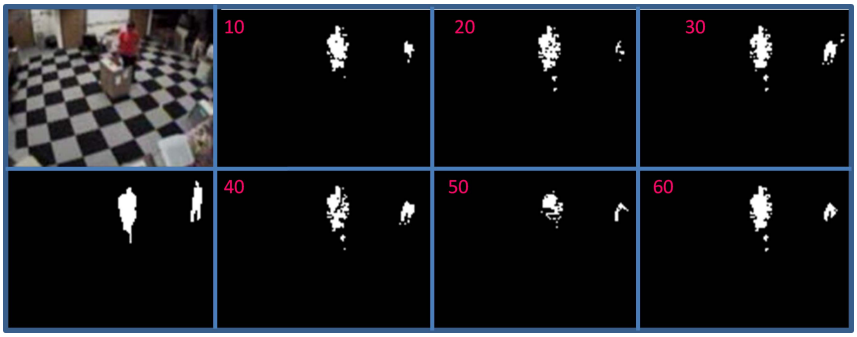


Figure 2.11. Tests for different values of N on video 4. From left to right and top to bottom: (i) image from the original video, (v) presents the hypothetical output for an ideal and perfect segmentation, and (ii), (iii), (iv), (vi), (vii) and (viii) correspond to the resulting video for $N = 10$, $N = 20$, $N = 30$, $N = 60$, $N = 50$ and $N = 40$ respectively.

The value of N, Number of frames for the update of the background model, depends on the frames per second available in the recorded scene. The best results are obtained for values of N that correspond from 1 to 2 updates per second. A new set of tests was carried out for values of 20, 30 and 40 for parameter N in videos 1, 2 and 3. Figure 2.12 shows the results obtained for a certain frame of each video. Table 2.7 presents the time consumed with respect to the time consumed by MGM and the ratio of false-positives (FP), false-negatives (FN) and successfully classified pixels.

Taking into account the results of Table 2.7, the increase/decrease of time and error ratios can be computed to obtain more significant results. The increase I_p from a value p_{MGM} is MGM to a value p_{MMGA} in MMGA of a parameter p is calculated as $I_p = \frac{p_{MMGA} - p_{MGM}}{p_{MGM}}$. It can then be obtained that a 50% increment of the value of N from 20 to 30 causes the decrease of execution time by 8.28% for video 1, 19.48% for video 2 and 5.36% for video 3. A further increment of N by 33%, from 30 to 40, results in a reduction of execution time of 7.77% for video 1, 32.65% for video 2 and 1.85% for

Table 2.7. Time consumed with different values of N compared to MGM (II)

Video	N	Time consumed	False-positive ratio	False-negative ratio	Success ratio
Video 1	20	57.09%	0.29%	1.75%	97.96%
	30	52.36%	0.27%	1.65%	98.08%
	40	48.29%	0.21%	1.77%	98.02%
Video 2	20	78.38%	0.12%	0.22%	99.66%
	30	63.11%	0.13%	0.23%	99.64%
	40	42.50%	0.15%	0.23%	99.63%
Video 3	20	45.78%	0.13%	1.37%	98.49%
	30	43.21%	0.11%	1.43%	98.46%
	40	42.41%	0.12%	1.42%	98.45%

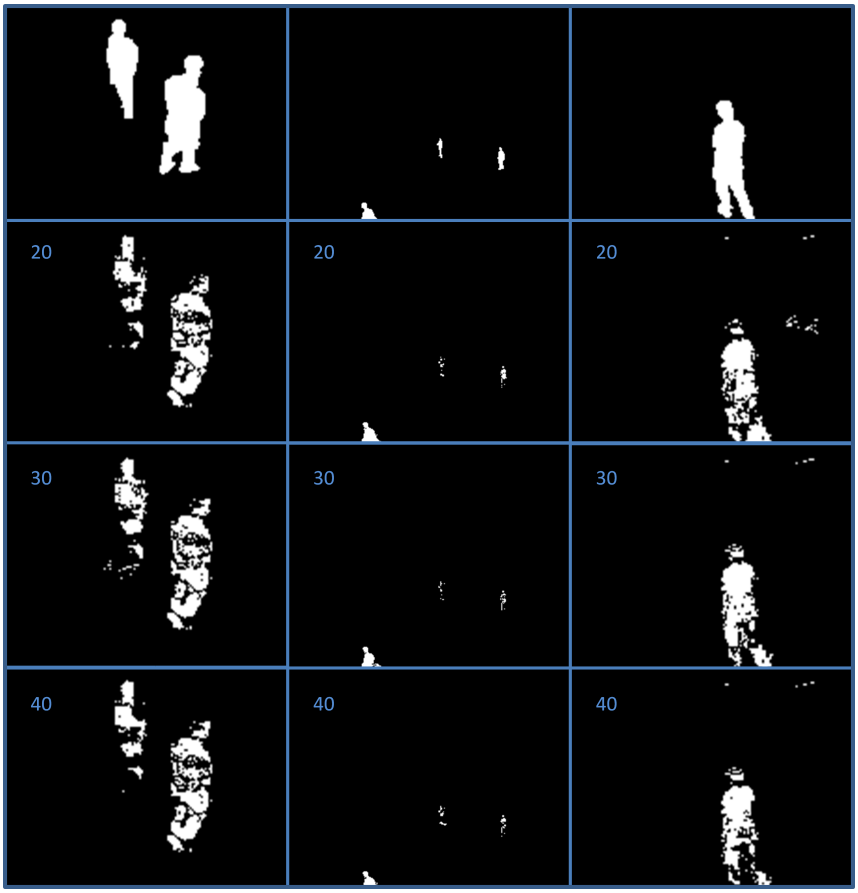


Figure 2.12. Tests for different values of N on videos 1,2 and 3. Left column: video 1, middle column: video 2 and right column: video 3. For the three columns, top to bottom: (i) image of the ideal segmentation, (ii), (iii) and (iv), correspond to the resulting segmentation for $N = 20$, $N = 30$ and $N = 40$ respectively.

video 3. As expected, increasing the number of frames considered for the update of the

model reduces the execution time, to a higher extent for longer videos, which makes the proposed algorithm especially advantageous for continuous recording systems.

Regarding total error, considering total error as the sum of false-positives and false-negatives, when N is increased from 20 to 30, the total error undergoes a decrease of 0.12% for video 1, and an increase of 0.02% and 0.03% for videos 2 and 3 respectively. Changing N from 30 to 40 has the effect of an error increase of 3.17%, 0.01% and 0.01% for videos 1, 2 and 3 respectively. The surprising results obtained for video 1 may be explained by the very strong changes in lighting in this video, which cause difficulties for the model to adapt itself to the background, resulting in a high rate of false-positives. This effect is mitigated for higher values of N ($N = 30$) thanks to the averaging of a higher number of frames for the update, until at a certain point ($N = 40$) no further improvement is achieved.

In the light of the results of the tests presented above, it is recommendable to use values of N that result in 0.75 to 1.25 update processes per second, setting $N = 30$, which seems to be a very reasonable value for videos with a frame rate of $15 - 30$ fps.

2.7.2.2 Performance evaluation

The main objective of the presented algorithm is to achieve a reduction in execution time without compromising the reliability and robustness of segmentation. Therefore, the evaluation of the performance of the proposed algorithm will focus on this twofold objective, execution speed and accuracy of the segmentation obtained, based on failure rates according to the number of background pixels classified as foreground, called false-positives, and the number of foreground pixels considered to be part of the background, known as false-negatives.

A comparison between the proposed algorithm and its predecessor, the MGM, has been established to assess the improvement achieved. The tests run on videos 1, 2 and 3 are presented in Table 2.8, and an image of each video is shown in Figures 2.13 to 2.15.

Table 2.8. Time consumed with different values of N compared to MGM (II)

Video	Execution time decrease	False-positive ratio variation	False-negative ratio variation	Overall reliability increase
Video 1	47.64%	-0.66%	0.20%	0.34%
Video 2	36.89%	-2.69%	0.15%	2.62%
Video 3	56.79%	-1.16%	0.75%	0.39%

The tests show that the proposed algorithm achieves a very significant reduction of execution time compared with MGM. Although the results show an increase of the false-negative ratio, these false-negative pixels are usually located within detectable blobs so that the errors can be easily resolved. Post-process modifications in the image are a simple solution to this problem, such as the close algorithm, widely used in

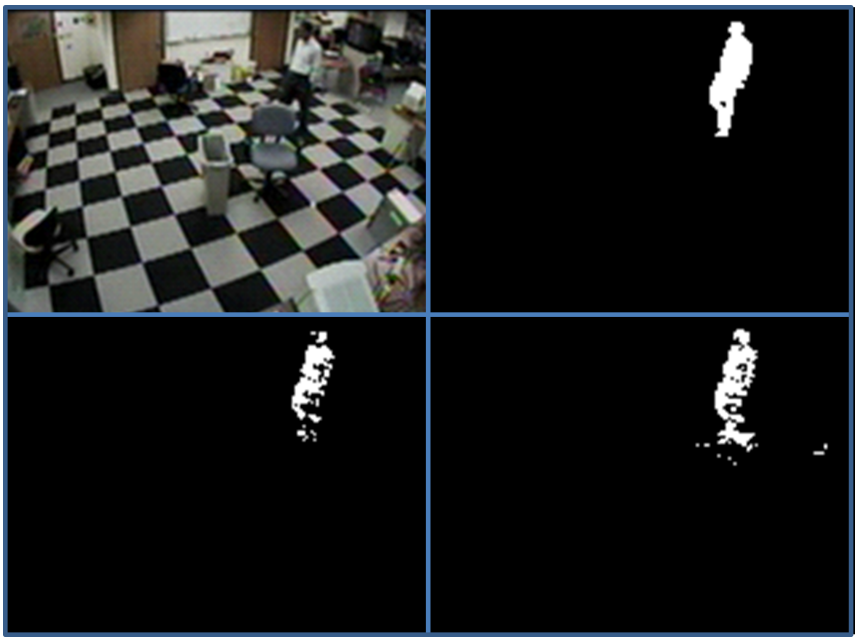


Figure 2.13. Test results on video 1. Upper row: image of video 1 (left) and ideal segmentation (right). Lower row: segmentation obtained with the MGM+RTDENN (left) and the MGM (right) algorithms.

Figure 2.14. Test results on video 2. Upper row: image of video 1 (left) and ideal segmentation (right). Lower row: segmentation obtained with the MGM+RTDENN (left) and the MGM (right) algorithms.

many computer vision applications [Yang et al., 2005]. Additionally, wrong pixels inside a detected blob can be ignored by many tracking systems. In contrast, the false-positive ratio is considerably reduced, so that the proposed algorithm provides an overall improvement in reliability, with a higher degree of robustness against noise and changes in lighting.

Figure 2.15. Test results on video 3. Upper row: image of video 1 (left) and ideal segmentation (right). Lower row: segmentation obtained with the MGM+RTDENN (left) and the MGM (right) algorithms.

2.7.3 Results in change detection dataset

Taking into account the performance evaluation carried out to both SDGM and MMGA, the final parameter values chosen are $F = 75$ for SDGM, and $v_{min} = 5$ and $N = 30$ for MMGA. The values of MGM parameters, shared with the two presented models, are the ones proposed by Power et al. [Power and Schoonees, 2002], that is: $\lambda = 2.5$, $T_B = 0.7$, $K = 3$ and $\alpha = 0.005$.

Goyette et al. [Goyette et al., 2012] presents a complete database to analyze change detection algorithms. This database covers a wide range of images and common difficult cases that any motion detection system, such as the one herein presented, has to face. Table 2.9 shows the time consumed by MGM, SDGM and MMGA, highlighting the most and least time-efficient results.

It can be clearly seen that both SDGM and MMGA perform faster than MGM in every single video sample. Nonetheless, a drawback of these techniques is that the error rate is slightly higher than in MGM. Tables 2.10 to 2.12 detail recall for each category; overall false positive and false negative ratios; overall specificity, percentage of wrong classifications (PWC) and precision, respectively. A reminder of how to compute all these evaluation methods is available in Appendix A.

In Table 2.11 it can be seen that although they reduce MGM's false-positive ratio, both new models incur in a greater ratio of false-negatives. Thus, SDGM and MMGA obtain poorer results in recall calculation, while having a better performance with regards to specificity. However, in every model, including MGM, the FN and FP errors

Table 2.9. Time (in seconds) consumed per model in each video

Category	Video	MGM	SDGM	MMGA
Baseline	Highway	332.574	**50.477**	105.309
	Office	424.185	147.968	**117.875**
	Pedestrians	177.905	**43.067**	95.754
	PETS2006	1215.96	**360.218**	453.596
Camera Jitter	Badminton	960.384	369.465	**340.908**
	Boulevard	409.715	364.945	**188.588**
	Sidewalk	228.974	142.098	**104.205**
	Traffic	217.529	208.887	**119.512**
Dynamic Background	Boats	1057.37	855.426	**454.671**
	Canoe	198.071	150.54	**107.955**
	Fall	3117.63	1348.9	**1004.49**
	Fountain01	255.054	**124.391**	176.492
	Fountain02	322.052	**146.333**	168.482
	Overpass	450.344	360.382	**181.041**
Intermittent Object Motion	AbandonedBox	1278.57	849.618	**469.485**
	Parking	362.984	247.122	**157.639**
	Sofa	473.522	**150.25**	204.089
	StreetLight	421.136	**112.708**	197.407
	Tramstop	1054.13	822.671	**375.92**
	WinterDriveway	325.065	276.542	**167.133**
Shadow	Backdoor	276.431	130.95	**118.048**
	Bungalows	314.698	159.235	**129.178**
	Bus Station	202.106	132.184	**114.891**
	Copy Machine	2654.6	1714.09	**852.246**
	Cubicle	1422.66	654.293	**598.345**
	People In Shade	249.014	**93.1951**	116.891
Thermal	Corridor	701.82	527.058	**303.13**
	Dinning Room	676.498	436.816	**213.254**
	Lake Side	1093.61	**360.875**	448.148
	Library	977.856	**323.606**	364.727
	Park	142.332	92.3611	**46.837**

Table 2.10. Recall per model in each video

Category	MGM	SDGM	MMGA
baseline	0.8498671112	0.6786037840	0.7478993647
cameraJ..	0.7538967965	0.7142748731	0.6254287367
dynamic..	0.8248719531	0.6578902766	0.8086573923
intermi..	0.6555188904	0.5534205858	0.5167141330
shadow	0.8278813070	0.6391029168	0.7382771694
thermal	0.5620764546	0.5204407751	0.6047094554
Overall	0.7456854188	0.6272888686	0.6736143753

Table 2.11. FP and FN per model in each video

Category	MGM	SDGM	MMGA
FPR	0.0453983031	0.0386376028	0.0355327095
FNR	0.0150097825	0.0198803370	0.0214736379

Table 2.12. Other measurements per model in each video

Category	MGM	SDGM	MMGA
Specificity	0.9546016969	0.9613623972	0.9644672905
PWC	5.5513135647	5.3506477307	5.2938549437
Precision	0.5636170846	0.5803588878	0.4955230634

remain relatively low and acceptable for security applications. Moreover, the number of wrongly classified pixels is slightly lower in the presented models.

2.8 Conclusion

Background subtraction is one of the key steps in motion detection using static video cameras. New models called SDGM and MMGA that could be used with any background subtraction technique have been implemented and described. MGM was selected to illustrate the performance of both algorithms, since it has been proved to be one of the most efficient models in this field and the common comparison algorithm used in several studies.

Results show three advantages of the SDGM. Firstly, the model's performance is not strongly dependent on its unique parameter F within a large range of values, meaning that choosing a parameter is not crucial for execution. Secondly, the model keeps the motion detection error ratio at nearly the same level as MGM. Finally, the SDGM significantly improves on the time performance of the background subtraction model. As shown in the test results, it consumes on average approximately a third of what MGM consumes, for any value of F.

However, the model may have some disadvantages as well. If the parameter F is extremely badly chosen, the model will increase the total number of wrongly classified pixels, introducing false-negatives. For this reason values of F lower than 3 seconds are recommendable. In any case, the time taken is considerably reduced. It is worth using SDGM in scenes where moving objects occupy up to 40% of the image, and it may be used in more crowded scenes. It drastically reduces computational time of the background subtraction algorithm, and keeps the error ratio low.

The MMGA is reliable, fast and adaptable to changes in the environment. The algorithm achieves an important improvement of both execution speed and segmentation

accuracy with respect to other currently implemented methods, reducing execution times by up to 57% and incrementing reliability rates by up to 2.5% in comparison with the MGM algorithm. It is the recommended background subtraction algorithm to use in cases where the information needed by the SDGM is not available.

This chapter has presented several contributions of this thesis. First, a new mathematical model describing what has been called gates was designed. Secondly, the Static and Dynamic Gates Model, the application of gates to background subtraction algorithms, was developed, and a great reduction in time consumption was obtained. Thirdly, the RTDNN algorithm, originally used in robotics, was applied in the MMGM; this novel application of RTDNN was implemented to efficiently update the background model in the background subtraction algorithms. Both SDGM and MMGA contribute greatly to fulfillment of the objective: reduction of the computing time needed in the motion detection step of a security system to make it run behavior analysis models on a real-time basis.

The creation of an autonomous video surveillance system will require the implementation of subsequent processes. After extracting the foreground, the next phases will include the tracking and identification of moving elements in the foreground, the detection of any dangerous situations or threats that may arise, alarm management and reporting to agents such as clients or security forces. Nevertheless, an efficient background/foreground segmentation is critical, while also leaving resources unused for subsequent steps. This is where the importance of fast algorithms such as those presented in this thesis lies.

Chapter 3

Object Tracking

3.1 Introduction

Video analysis is composed of three key steps: detection of interesting moving objects, studied in the previous chapter; tracking these objects from frame to frame; and finally the analysis and understanding of the observed tracks to recognize the behavior of the objects. This chapter addresses the problem of interesting objects tracking. There are many different approaches to this problem where, commonly, the output of a motion detection algorithm is used as the input of tracking systems.

Tracking algorithms are used to distinguish the movements of interesting detected objects in a scene. It is essential in several fields such as traffic transportation [Tai et al., 2004], video compression [Sikora, 1997], human-machine interfaces [Liu et al., 2012] and, of course, security and defense.

This thesis presents a probabilistic tracking algorithm. In the first place, an approach to distinguish color pixels from gray ones is described. That way, each blob obtained from the motion detection will be modeled with two adaptive histograms, one for color and one for gray pixels, in addition to the current position, speed and direction. These histograms are used to calculate the color distance to previous tracked objects, in order to identify to which object each blob belongs. Secondly, the tracking problem is broken down into six cases, depending on the proximity between currently detected movement (blobs) and previously tracked objects. The matching between both blobs and objects is carried out automatically in certain simple cases to speed up the algorithm, while harder cases use a more robust model, such as a color appearance model based on color and gray histograms. Finally, a probabilistic model based on *segments* is introduced to track occluded objects and to correct previous uncertainties, as well as to facilitate the behavior understanding step that modern security systems are provided with. Each detected object has a set of possible trajectories with different probabilities throughout its lifetime, and the most probable one is chosen to feed the behavior analysis module.

In this chapter, Section 3.2 gives a review of previous works done. The color model used is explained in Section 3.3, while Section 3.4 details problem decomposition cases and their solutions. Afterwards, Section 3.5 presents the proposed methodology, introducing the *segments*, which are essential to the probabilistic model. Next, the results are shown and, finally, the last section contains the conclusions reached.

3.2 Previous works

Visual object tracking is a challenging problem that has attracted the interest of the research community. In recent years, several published reviews have dealt with this problem. On the one hand, some of these studies are very application-focused. In [Aggarwal and Cai, 1999, Gavrila, 1999, Moeslund and Granum, 2001, Forsyth et al., 2005], the authors analyze the tracking of persons, where the periodic movement of legs may be used to easily track people. Zhan et al. [Zhan et al., 2008] focus their research on crowd analysis, which is necessary in certain security environments where individual tracking, and thus individual behavior recognition, is impossible. Tracking from on-board cameras is another attractive area, for tracking both other vehicles [Sun et al., 2006] and pedestrians [Geronimo et al., 2010].

On the other hand, several surveys present solutions for broader problems, without taking anything for granted about the input images or the environment where tracking takes place [Wang et al., 2003, Hu et al., 2004a, Yilmaz et al., 2006, Cannons, 2008, Ko, 2008, Li et al., 2013]. Many of these studies not only tackle how to robustly track interesting objects, but also cover other closely related areas such as motion detection or behavior analysis. Therefore, the work presented in these surveys seems more interesting for the objective of this thesis.

There are two stages to obtaining the whole trajectory of an interesting object to be tracked: deciding which features are used to model the tracked object and tracking the object thanks to these features. The following subsections detail each of these stages.

3.2.1 Feature selection

In first place, a suitable feature for tracking has to be chosen. Feature selection is a common stage in several computer vision applications. Boal et al. [Boal et al., 2014] review the standard techniques, of which the Scale Invariant Feature Transform (SIFT) [Lowe, 2004] is one of the most widely used. However, the main disadvantage of this technique is the computational power it requires. Bay et al. [Bay et al., 2006, Bay et al., 2008] implement a speed-up algorithm to try to solve this bottleneck. They greatly reduce the computation time compared to SIFT, making SURF an ideal technique in object recognition applications. Even though they reduce resource consumption, these

feature selection techniques remain too slow to be used in real-time applications such as security systems, because they were intended to be used in off-line pattern recognition applications.

Among the most common feature selection techniques used in surveillance and security applications are boundaries, optical flow and color.

The simplest boundaries a system can use as feature are edges, as in the Canny detector [Canny, 1986] which has proved to be a robust edge detector [Bowyer et al., 1999, Bowyer et al., 2001]. However, this technique cannot correctly track non-rigid objects. Active contour representation [Paragios and Deriche, 2000, Cremers, 2006, Sun et al., 2011] allows robust tracking of non-rigid objects using their boundaries, although this increments the consumption of computing resources.

Optical flow models the movement of each pixel in the scene through a dense field of displacement vectors. Barron et al. [Barron et al., 1992, Barron et al., 1994] and Liu et al. [Liu et al., 1998] present reviews of optical flow performances. Lucas and Kanade [Lucas and Kanade, 1981] is one of the most commonly used techniques.

Many different color spaces have been used in computer vision applications. A thorough analysis of different spaces was first made in [Ohta et al., 1980] in search of a color space for segmentation. Currently, the most common space is RGB, but it does not correspond to how human eyes see [Paschos, 2001, Yilmaz et al., 2006]. Paschos studies $L^*a^*b^*$ and HSV color spaces to conclude that both perform better than RGB, and that HSV works even better than $L^*a^*b^*$. Nummiaro et al. [Nummiaro et al., 2003] also use HSV to make the algorithm less sensitive to lighting conditions.

Not only is color the most broadly used classifier, but it is also the classifier that requires the least post-processing. Thus, color is the ideal feature to use in the case where execution time consumption is a major constraint, but the silhouette of the object to be tracked can be modified, such as surveillance systems, where, generally, the tracking algorithm has to deal with non-rigid object, such as people, and in the fastest way possible. Color information is usually stored in the form of histograms.

3.2.2 Tracking methods

There is a classification of the most used tracking methods in [Yilmaz et al., 2006]. The authors present three main categories: kernel tracking, silhouette tracking and point tracking.

Kernel tracking uses the object representations to track them by computing the motion of the kernel in successive frames. Contrariwise, silhouette tracking estimates the real object shape in every frame. It makes intensive use of either edge detection, shape matching or contour evolution techniques. Point tracking is used to track small objects that are simply represented by points, or to locate interesting points in a scene

for localization purposes. The tracker essentially uses the position and motion of the points in the previous frames. Although interesting in many applications [Veenman et al., 2001], point trackers are not used in security systems as there is usually enough information about the size and shape of the object to be tracked in those systems.

Template based techniques are the main kernel tracking techniques. In a template matching system, the algorithm looks in the whole image for a region similar to the templates available. As explained in the previous chapter, the main flaw of this technique is intense resource consumption. Various solutions can be adopted to diminish this consumption. One solution is to reduce the search to certain zones, generally near the last known position of the tracked object. Fieguth et al. propose performing the search of the tracked object, in their case human heads, in the eight adjacent positions to the predicted location of the object [Fieguth and Terzopoulos, 1997]. In [Santner et al., 2010], the authors present a robust technique to track objects that combines the template information of the tracked object, obtained manually in the first frame, with optical flow and an appearance-based tracker.

Several researchers have used other less demanding object representations instead of templates, such as color histograms. Color histograms are computed using the pixels inside the kernel (usually a rectangle or an ellipse). Color histograms are widely used in tracking modules to identify different objects in the scene and to tell them apart. Nummiaro et al. [Nummiaro et al., 2003] propose a particle filter model using histograms. The particle filter can handle multiple hypotheses that permit tracking of multiple objects even in scenarios with occlusions. Color distribution provides information about the object being searched for, but the object must be found in each new frame. Comaniciu et al. [Comaniciu et al., 2003] present a mean-shift approach to reduce the search space and the computing time compared to a brute force template matching algorithm, and then use histograms as a matching feature. In a similar way, a mean-shift tracker is used in [Leichter et al., 2010]. The authors propose a technique using multiple reference color histograms. Each one of these histograms models a different view of the object which has to be available prior to tracking.

Nevertheless, mean-shift-based approaches may lose tracking due to occlusions [Khan et al., 2011]. Khan et al. propose the use of a combination of particle filter and Mean Shift, to reduce the computing time of particle filter as well as to improve mean-shift solutions to occluded objects, although in crowded scenes the computation becomes critically slow. Li et al. point out that mean-shift trackers do not run correctly in occlusions. In [Li et al., 2008], they propose a technique using layers to represent a single object in each layer allowing occluded objects to be in different layers, behind detected objects, improving the mean-shift algorithm.

A layering technique is also introduced in [Tao et al., 2002]. The authors propose a new model to cope with drifting or mismatching due to similar appearances. The shape (boundaries), appearance (color) and motion (speed) of the foreground objects,

obtained after a layering decomposition, are analyzed. An EM algorithm, which works with videos obtained from cameras with zoom, pan and tilt, is used to track the objects. However, although robust against stationary objects or close-by objects, it does not correctly track articulated objects such as people.

Finally, silhouette tracking can be developed through shape matching or contour tracking. The W^4 surveillance system presented in [Haritaoglu et al., 2000] tracks objects assuming constant velocity and then refines this information thanks to a correlation of the object edges inside its silhouette in consecutive frames. As it does not use color information, this shape matching technique runs in grayscale videos. In contour tracking techniques, Cremers et al. apply optical flow to the image to constrain the contour evolution algorithm so that it can only contain pixels with similar motion [Cremers and Schnörr, 2003]. Silhouette tracking is usually the reference technique when the application needs the whole shape of the tracked object in very specific applications, but not in security systems where fast algorithms are required.

A novel tracking model, using color space model described in next section, that copes with occlusions, tracks articulated objects and works in real time is presented in section 3.5.

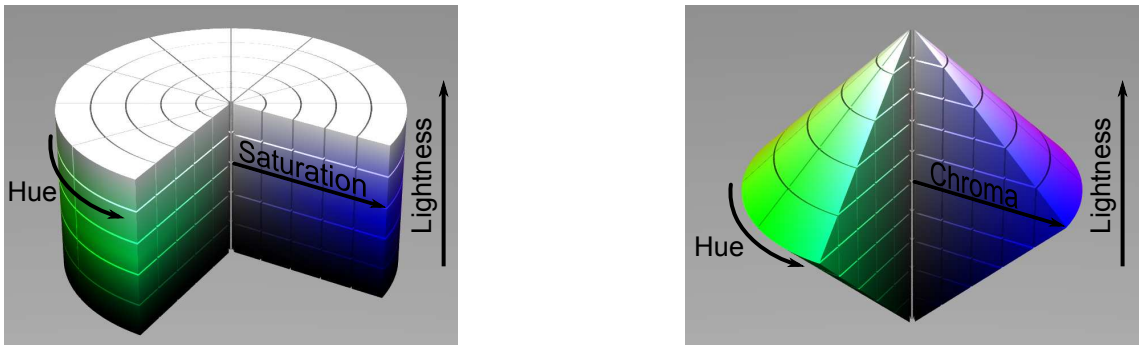
3.3 Color model

3.3.1 Color space

The research herein presented employs HSL color space, which is similar to HSV. As pointed out by Nummiaro in [Nummiaro et al., 2003], changes in illumination can modify the RGB color value, thus it is important to use the hue channel to identify pixel color. However, too dark, too bright or gray pixels may be wrongly classified if only the hue channel is observed. To avoid this problem, this thesis proposes the separation of color classification into two categories: gray pixels and color pixels.

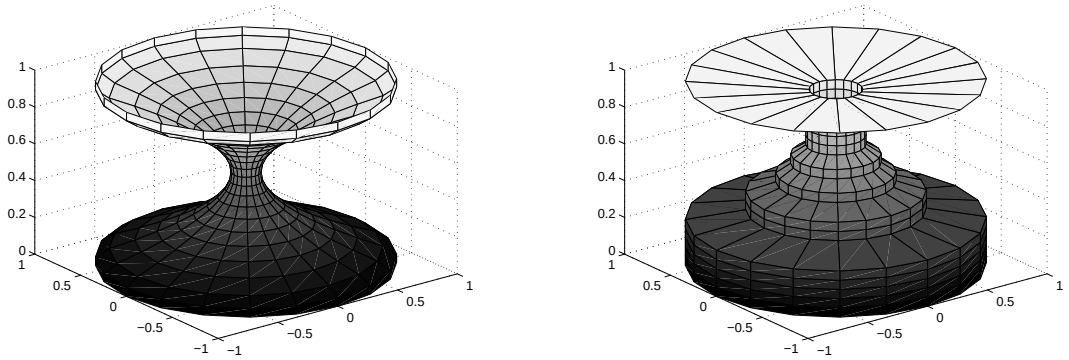
If the chroma channel is used instead of the saturation channel, it is only necessary to take the “peaks” of the bottom and top cones as well as the central cylinder as grayscale pixels, while the rest remain as colorful pixels (see Figure 3.1). In the HSL color space cylinder, the top and the bottom “peak” cones become cylinders, and the central cylinder takes a more complex shape (see Figure 3.2a).

To make it easier to program, the central cylinder is decomposed into several simple cylinders, similarly to what is done in [Tseng and Chang, 1992], as shown in Figure 3.2b. Tseng and Chang propose a set of thresholds that are largely more restrictive in dark regions than in bright ones because dark pixels have a higher uncertainty than bright pixels [Kim et al., 2005].



(a) HSL **(b)** Hue-Chroma-Lightness

Figure 3.1. HSL color space (a) and Hue-Chroma-Lightness color space (b).



(a) Theoretical approach **(b)** Tseng approximation

Figure 3.2. Theoretical (a) and Tseng approximation (b) border separating gray and color zone of HSL color space.

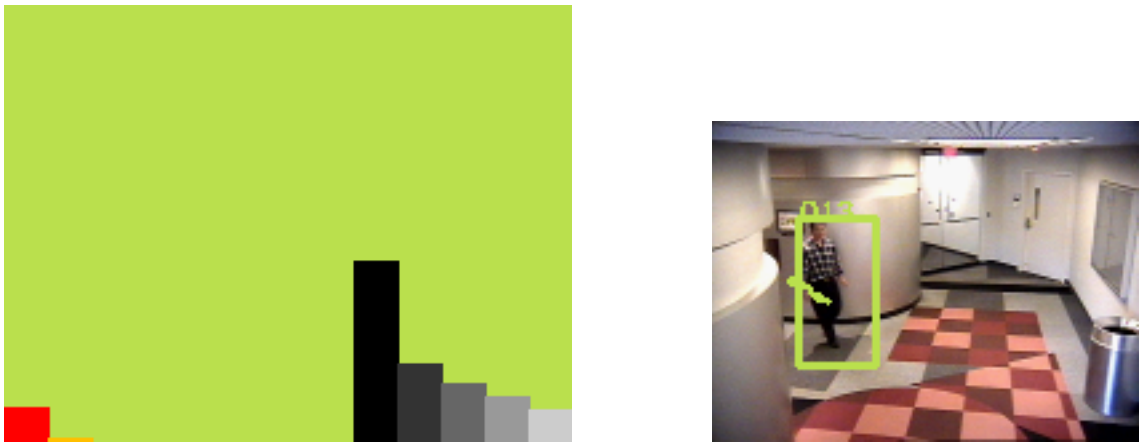
Once split, either the hue channel for color pixels or the lightness channel for gray pixels gives the color information needed. For each pixel, only one component of the $3D$ vector modeling the color is used.

3.3.2 Tracked object modeling

As explained, objects are tracked to know the trajectory each person describes in a scene in order to analyze the scene in search of dangerous situations. A color model is used to identify each object. Once the background subtraction algorithm provides the pixels the blob is composed of, they are separated in two categories: color pixels and gray pixels, as seen in Section 3.3.1. A histogram is created for each category, taking the hue value in color pixels and lightness value in gray pixels.

In general, colorful pixels will carry the greater part of the information, so the color histogram will generally be modeled with more bins than the grayscale histogram. These two histograms model a blob, providing the needed information to the tracking system.

To make the calculation of the color distance possible (see Section 3.3.3), the histograms must be normalized as a unique histogram with a number of bins equal to the sum of the bins of both histograms. Figure 3.3 shows an example of a histogram.



(a) **(b)**

Figure 3.3. Example of a histogram containing both color and grayscale values (a) and the person it models (b).

3.3.3 Color distance

To know if the present blob models the same person or group of persons and represents a new position of a previously tracked person or group of persons (from now on, the *object*), the distance between the histograms of the present blob and past object is calculated. Generally, to compute the distance between two histograms h^1 and h^2 of n bins, the Bhattacharyya coefficient $BC(h^1, h^2)$ [Kailath, 1967] is calculated as:

$$BC(h^1, h^2) = \sum_{k=1}^{n} \sqrt{h^1(k) h^2(k)} \tag{3.1}$$

Then, the Hellinger distance d of both histograms is obtained as follows:

$$d = \sqrt{1 - BC(h^1, h^2)} \tag{3.2}$$

The Hellinger distance is used because other distances such as the Bhattacharyya distance violate one of the axioms of metrics [Comaniciu et al., 2003], namely, the Bhattacharyya distance does not satisfy the triangle inequality [Kailath, 1967] which states that, in a triangle XYZ, the sum of the lengths of any two sides must be greater than or equal to the length of the remaining side: $\overline{XY} + \overline{YZ} \geq \overline{XZ}$.

From Equations (3.1) and (3.2), it can be deduced that the final distance $d(b, o)$ of a blob b and an object o, knowing the color Bhattacharyya coefficient BC_C and gray

Bhattacharyya coefficient BC_G, is calculated as (see Appendix B for more details):

$$d(b,o) = \sqrt{BC_C + BC_G - 1} \tag{3.3}$$

where BC_C and BC_G are the Bhattacharyya coefficients between color and gray histograms respectively. A threshold T_d is set to indicate the maximum acceptable color distance between an object and a blob to be considered as modeling the same object.

3.4 Problem decomposition in cases

The proposed tracking methodology uses the color data described in the previous section, the position information of each object in previous frames and the objects' proximity to current frame detected blobs. First, the model groups adjacent blobs and objects. These groups are then classified in one of the six different cases into which the tracking problem has been decomposed, depending on the number of blobs and objects they contain. In each case the model tries to match blobs and objects of the group. The cases are solved using a fast straightforward solution for easy common cases, and a more robust and complex matching solution for the rest. This section presents the six different cases into which any tracking situation may be classified and expounds how to solve each of these cases using the information available.

3.4.1 Cases

As has been seen in Chapter 2, background subtraction algorithms segment the image into background and foreground pixels. A set of blobs can be obtained from the binary image generated. These blobs are not an accurate estimation of people in the scene, as they can contain more than one person or, conversely, a blob may just be a limb, and should be added to others to obtain a full body [Zhao and Nevatia, 2004]. Blobs are the new pixels in motion obtained from the motion detector in each frame, while objects are the actual people already recognized in previous frames, which the tracking model should match to one of the blobs in the present frame, so that the object's whole trajectory may be stored. The technique to track people from blobs herein proposed takes into account such disadvantages.

Each blob in each new frame is matched to an existing object. If a blob is not matched, it is associated to a newly created object. Conversely, each object not matched after several frames is considered to have disappeared. The objects are thus tracked knowing that each one can represent one or more people, similarly to [McKenna et al., 2000], such that occlusions may occur, and that groups of people may be formed. The

tracking stage provides useful information so an analysis of the behavior based on the trajectories and speeds of each analyzed object is made possible.

All the different difficult cases that the algorithm must face, where objects appear or disappear from frame to frame, are presented here (see Figure 3.4). A case is considered to be a problem that requires a solution using the tracking algorithm where some tracked objects and some new blobs meet in the scene. Depending on the number of blobs and objects, the model will face a different problem. It has to be taken into account that any object occluded under another is considered to be in the scene, so they are counted as two objects. Although any case is a special case of *case 6*, they are separated to make it clearer and easier to program.

(1) No objects, any number of blobs.

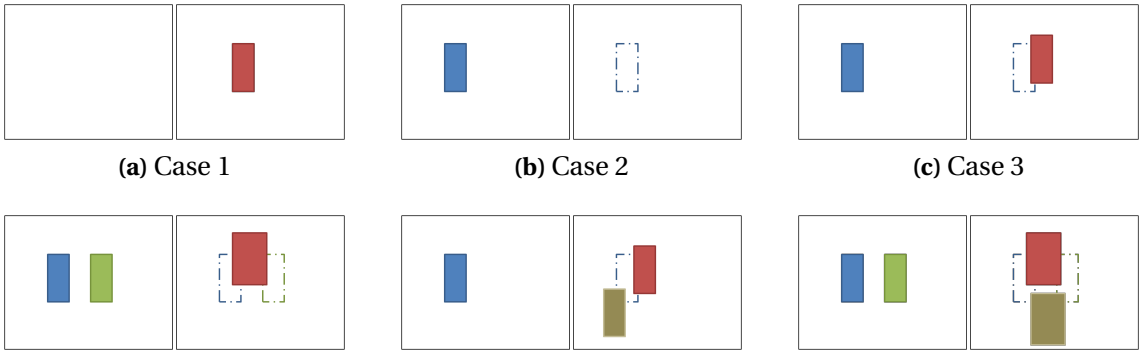
(2) No blobs, any number of objects.

(3) Just an object and a blob.

(4) Several objects and a blob.

(5) An object and several blobs.

(6) Several objects and blobs.



(a) Case 1 **(b)** Case 2 **(c)** Case 3

(d) Case 4 **(e)** Case 5 **(f)** Case 6

Figure 3.4. Set of six pairs of images, with bounding boxes representing objects in the previous frame (on the left), previous objects with a dotted line and blobs detected in the present frame with a solid line (on the right), for the six cases.

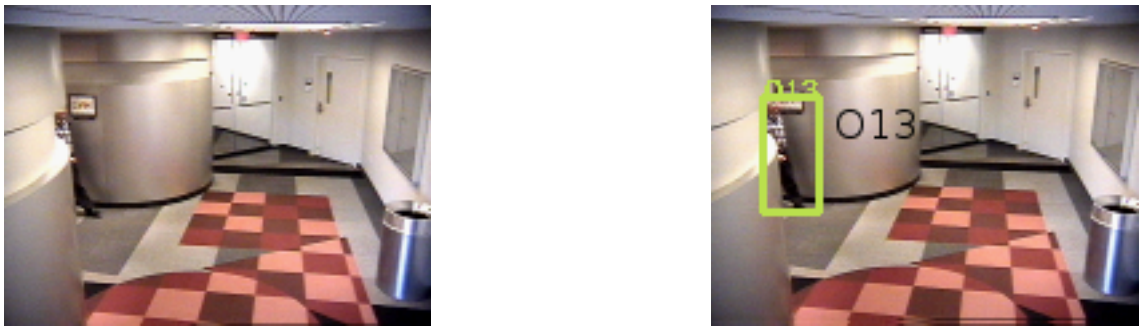
3.4.2 Solving cases

In order to tackle the problem of matching past tracked objects with present blobs, groups of blobs and objects are formed according to their proximity to each other. A group consists of a set of blobs and objects close to each other. They are considered to be close if their bounding boxes overlap, considering a bounding box to be the rectangle that completely envelops the detected blob. The assumption of overlapping bounding boxes is a valid assumption as security systems work in real time and thus, an interesting object does not have enough time to move far from its previous position from frame to frame. Each group has the structure of one of the presented cases, and it is solved

depending on the color distance between objects and blobs. The following subsections present an example of each case..

3.4.2.1 Case 1: No past objects, blobs detected.

This is a trivial case, where there were neither blobs nor objects in the past frame. The model creates a new object to be tracked for each blob in this case (see Figure 3.5).

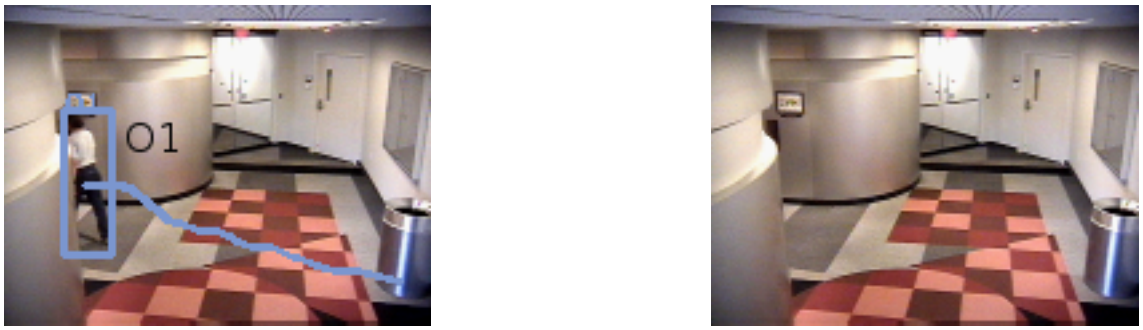


(a) **(b)**

Figure 3.5. Case 1: object $O13$ appears.

3.4.2.2 Case 2: No present blobs where previous objects were.

In this case, the tracked object has disappeared. It is recommended to maintain the object active in the model for several frames to confirm if it is a temporal total occlusion, or if the object has merely left the scene. See Figure 3.6 for an example.



(a) **(b)**

Figure 3.6. Case 2: object $O1$ disappears.

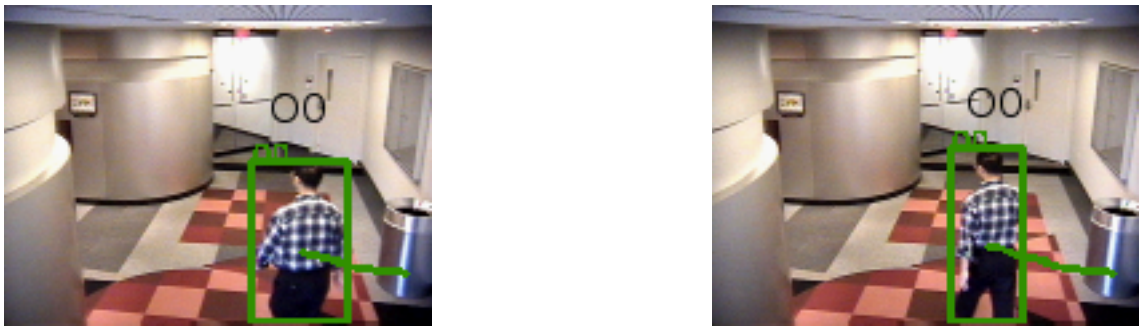
3.4.2.3 Case 3: A unique object and a unique blob.

If the color distance between the blob b and the object o is low enough (see Equation 3.4), it is a simple problem, where the object has followed its path without running into

any other object. On the contrary, if the distance is too wide it may indicate that the previous object has left the scene at the same place and time as a new object has entered it.

$$d(b,o) = \sqrt{BC_C + BC_G - 1} < T_d \tag{3.4}$$

T_d is the threshold that delimits what is considered to be a close color distance between a blob and an object and what it is not. Of course, if the information about the entrances to the scene is available, in a similar way to the static gates seen in 2.5, the color distance may only be used in such zones. That is, an object can only leave the scene through those entrances and thus this possibility is only accepted there. See Figure 3.7 for an example.



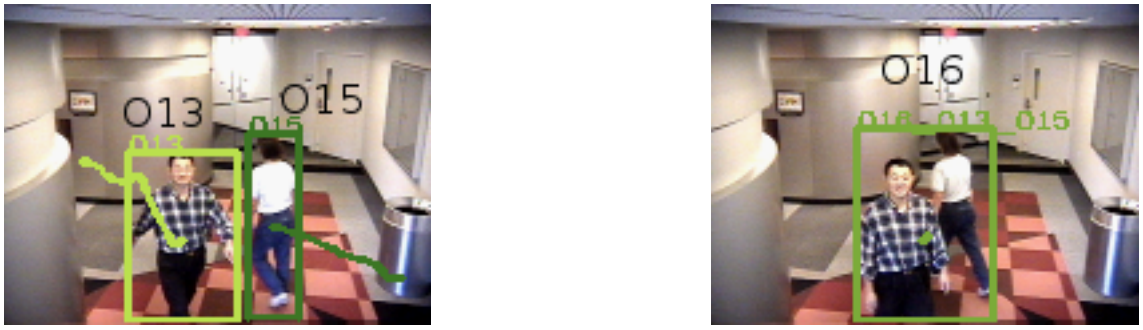
(a) **(b)**

Figure 3.7. Case 3: object *O*0 continues its movement.

3.4.2.4 Case 4: Several objects and a blob.

In this case, a single blob is detected where there should be several. The model considers that all the objects have met and travel together, occluding themselves. A new fictitious object that gathers the blob color information is created: it is considered that all of the objects that were present in the previous frame are occluded under this object. Figure 3.8 illustrates this case.

The newly created fictitious object is solely used as long as the object only takes part in *case 3s*. Once the object disappears, due to a *case 2*, all its information is deleted, only the information of its occluded objects is used to create trajectories to feed the behavior analysis step of the security system. If, before disappearing, the new object is grouped with several blobs, instead of resolving a *case 5* (*an object and several blobs*), the occluded objects take the place of the new fictitious object transforming the problem into a *case 6* (*several objects and several blobs*). These cases are explained next, and Section 3.5 details the probabilistic implications.

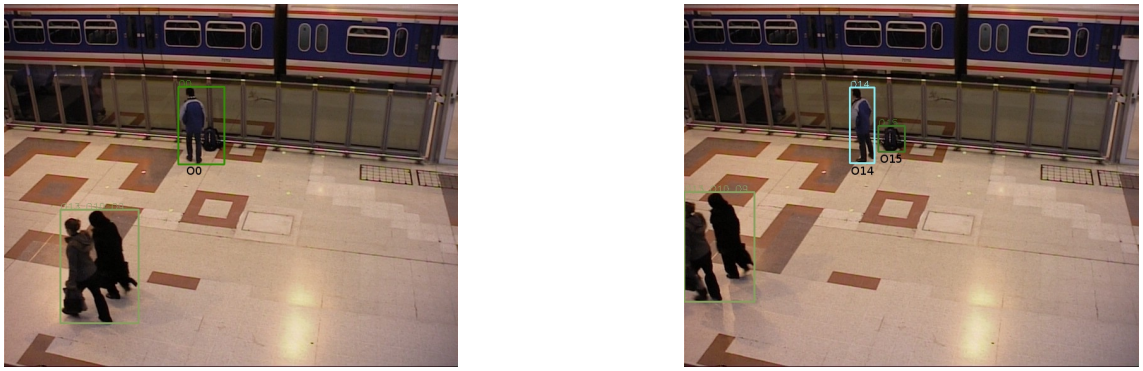


(a) **(b)**

Figure 3.8. Case 4: objects $O13$ and $O15$ merge in $O16$, where both objects are occluded.

3.4.2.5 Case 5: An object and several blobs.

Several blobs are detected where there was previously a single object. The model considers that the object is a union of several occluded interesting objects, and thus treats it as a *fictitious object*, that is, its information is not directly used in subsequent processes. However, the information of where it has been is kept. Each new blob creates a new object in the model, similarly to a *case 1* (see 3.4.2.1), but adds the past trajectory of the *fictitious object* to each new one. These newly created objects have not just appeared in the scene, they were occluded under the *fictitious object* and thus, they have to inherit the information of its past trajectory to be used in the behavior analysis step. Figure 3.9 shows a representation of this case.



(a) **(b)**

Figure 3.9. Case 5: object $O0$ splits in objects $O14$ and $O15$.

As previously explained, if the object was a fictitious object occluding other objects, the model would use those occluded objects and the problem would be treated as a *case 6* (see Subsection 3.4.2.6).

3.4.2.6 Case 6: Several objects and several blobs.

This case is the generalization of the previous ones. The color distance between objects and blobs is calculated, in order to create a matrix, similarly to the work in [Yang et al., 2005]. The matrix contains the value of the color distance between objects and blobs, where objects are ordered in columns and blobs in rows (see 3.10a for an example). Each iteration obtains the closest blobs to each object, 3.10b, and the closest objects to each blob, 3.10c. Any matching in both cases, that is, an object/blob couple where the blob is the nearest blob to the object and the object the nearest object to the blob, is then chosen as a couple and the row and column where it belonged are deleted, 3.10d. Being a couple means that the blob models the object in the present frame, and thus cannot model any other object (the row is deleted) and similarly, the object cannot be modeled by any other blob (the column is deleted). Of course, the couple is only an acceptable couple if the distance between the blob and the object is low enough (see Equation 3.4). If this condition is not met, the column and row are still deleted, but the object is considered to have disappeared, and the blob is considered to model a new object that has just entered the scene.

	O1	O2	O3
B1	6	3	4
B2	6	6	5
B3	7	5	8

(a)

	O1	O2	O3
B1	6	3	4
B2	6	6	5
B3	7	5	8

(b)

	O1	O2	O3
B1	6	3	4
B2	6	6	5
B3	7	5	8

(c)

	O1	O2	O3
B1	6	3	4
B2	6	6	5
B3	7	5	8

(d)

	O1	O3
B2	6	5
B3	7	8

(e)

	O1	O3
B2	6	5
B3	7	8

(f)

	O1
B3	7

(g)

Figure 3.10. Example of blob and object matching. $O1$ to $O3$ (in columns) are the objects of this example, $B1$ to $B3$ (in rows) the blobs. From top to bottom and left to right: (a) original matrix with distances; (b) closest blobs to each object highlighted; (c) closest objects to each blob highlighted; (d) chosen couple(s) in red and deleted rows and columns for the next iteration in gray; (e) matrix configuration of second iteration; (f) couple(s) chosen and deleted cells as in 3.10d; (g) last iteration.

If, at the end, there are no matched blobs remaining, a new object is created that inherits the trajectories of all the previously tracked objects. If, at the end, there are unmatched objects, they are occluded under the matched and newly created objects.

See Figure 3.11 that shows an example where two blobs appear where there were two objects ($O10$ and $O11$, occluded under $O12$).

(a) **(b)** **(c)**

Figure 3.11. Case 6 (from (b) to (c)), preceded by a case 4 (from (a) to (b)). The objects $O10$ and $O11$, previously occluded, are matched to the present blob and maintain their previous trajectories. A thin trajectory means that it is an estimation of where the object was during the occlusion.

3.5 Proposed methodology

The proposed tracking methodology uses the color data described in Section 3.3, and the position information of each object in previous frames as well as their proximity to current frame detected blobs. At first, as shown in 3.4, the model groups adjacent blobs and objects together. These groups are then classified in six different cases, depending on the number of blobs and objects they contain. Each case tries to match blobs and objects. Cases are solved taking this information into account, with a fast straight forward solution for easy common cases, and a more robust and complex matching solution in other cases. Afterwards the tracking system adds the information about the position of each object in this new frame and updates the probabilities of where each occluded object is and has been.

This section explains the probability model that takes place once a case is solved. First, *segments*, the heart of the probabilistic model, are introduced. This is followed by a description of how the final trajectory of an object is created and how the model can handle occlusions.

3.5.1 Segments

Behavior analysis and understanding requires information of past trajectories and the speed of the studied objects. This information is stored in what has been called *segments*. A segment s contains the information of a set of (x, y) coordinates, frame/time t and blob bounding box size. This information contains the basic elements needed in usual abnormal behavior detection models, such as [Xiang and Gong, 2008], gathered for

each detected object in each frame. Besides, a segment also contains the set of possible objects following that path at that time. Each of these objects has a probability p of being in that segment. If there is only one object, it will have a probability 1 of being there, and thus a probability 0 of being in any other segment of the image at that exact time. If the segment contains more than one object, a fictitious object is created containing the color and motion speed of the whole group of objects, but it will be deleted once the segment is no longer in use and its information is not used in the behavior analysis step. This object is considered to be the main object of the segment. If the segment only contains one object, it is also the main object of the segment. The main object of the segment carries the color and position information which is used to determine, in each frame, which of the six possible cases this group of objects belongs to. See Figure 3.12 for an example of segments.

If an object, at a certain time, can be located in more than one possible segment, then this object is considered to be occluded and it has a probability lower than 1 of being in any of the segments (see Figure 3.13a). If the sum of probabilities of an object of being in any segment at a certain time is lower than 1, the object may have disappeared. If an object o has a probability $P^{dis}(o,t)$ of having disappeared at time t, the accumulated probability $P^{dis}_{acc}(o,t)$ of the same object having disappeared before time t is the sum:

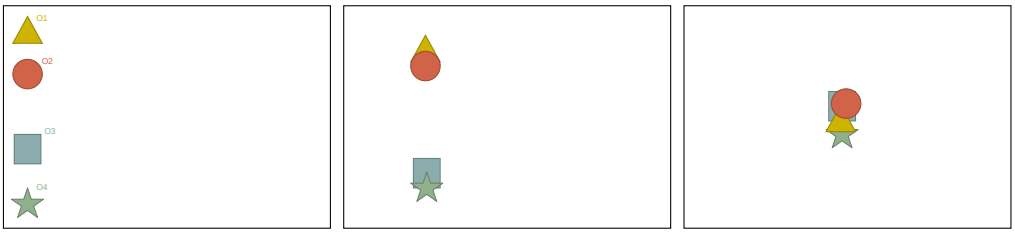
$$P^{dis}_{acc}(o,t) = \sum_{i<t} P^{dis}(o,i) \tag{3.5}$$

Additionally, the sum of the probabilities $P(o,s_i,t)$ of the total number $N_S(o,t)$ of segments s_i where an object o may have been at time t ensures:

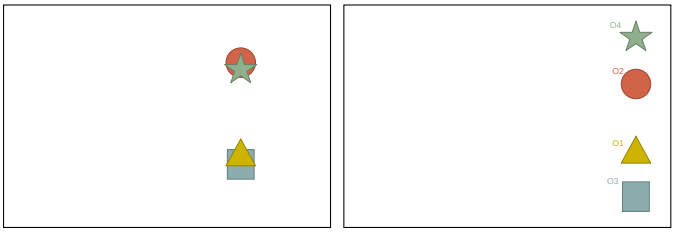
$$\sum_{i=1}^{N_S(o,t)} P(o,s_i,t) + P^{dis}_{acc}(o,t) = 1 \tag{3.6}$$

In each frame, each object belonging to a segment whose main object has been matched with a blob is updated with the newly gathered information. The color and grayscale histograms of the main object are also updated with the information of the new blob. Moreover, if a previously occluded object becomes visible again, the probabilities of all the previous segments are updated taking into account this information, as in Figure 3.13.

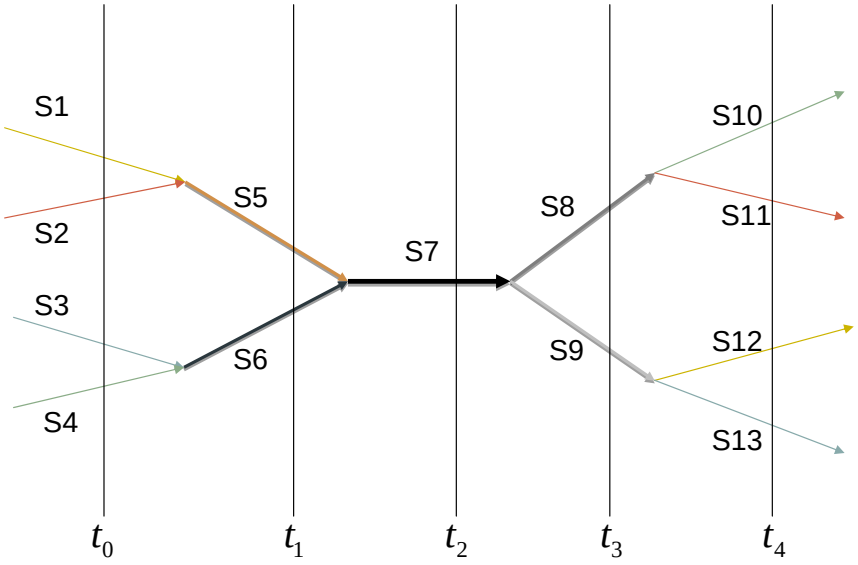
If a blob is matched with an occluded object (due to a *case 6* for example), this object becomes the main and unique object of the new segment, with a probability 1 of being in that segment. That means that this object has followed a trajectory that ends up in this segment. In other words, the object has to be eliminated from the sets of occluded objects of segments that do not lead to this one. The prior probability of being



(a) t_0 **(b)** t_1 **(c)** t_2

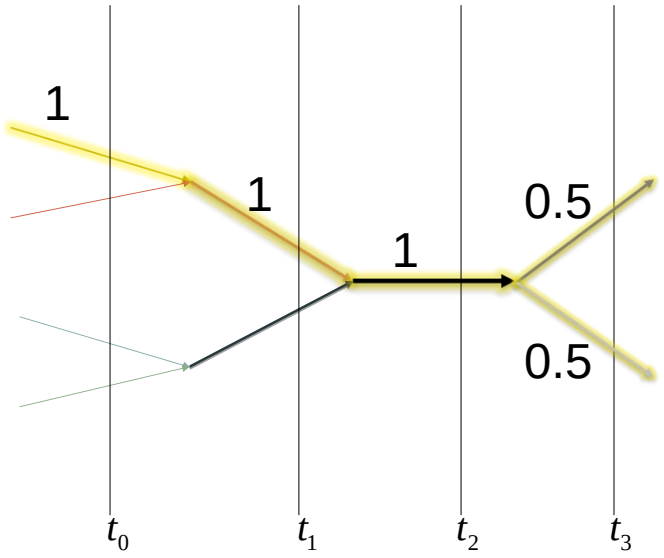


(d) t_3 **(e)** t_4

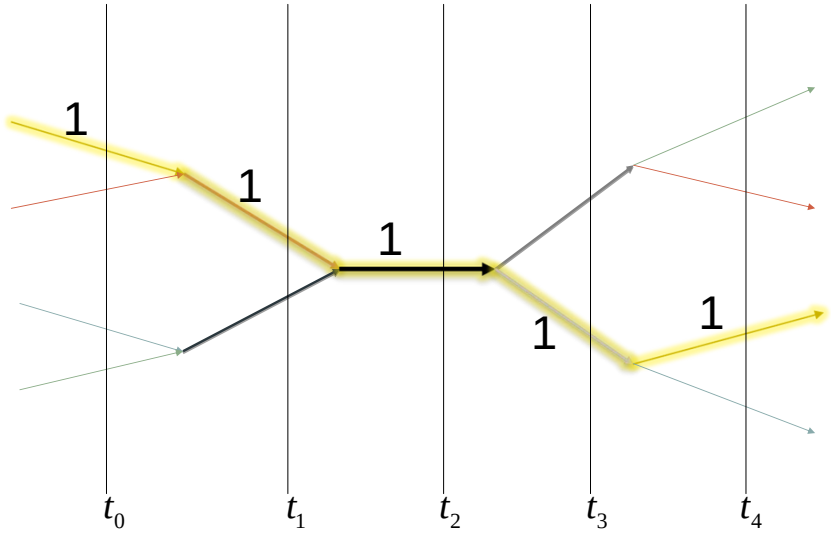


(f) Segments

Figure 3.12. Segment examples depicting the trajectories of four objects. Movements of objects, (a) to (e); segments obtained (f). Segment $S5$ only contains $O1$ and $O2$ with a probability of 1. At time t_3, segments $S8$ and $S9$ contain occluded objects $O1$ to $O4$ with a probability of 0.5 each. The model knows that, at time t_4, $S8$ did not contain object $O1$ and segment $S9$ contained it with a probability of 1.



(a) t_3



(b) t_4

Figure 3.13. Segments and probability of object $O1$ being in each segment at different times. Probabilities are updated backwards each time new information is collected.

in these segments has to be split between the segments that contain the object at the same time, which lead to the present segment (see Figure 3.12).

During the whole segment, the objects belonging to it and their probability do not change. If any of them changes, due to one of the cases explained in Section 3.4, a new segment is created and the old one is no longer in use. Every segment saves the information of the previous and following segments. This permits the creation of a set of trajectories with different probabilities of where and when each object has been, so that the different behaviors may be analyzed. The probability of a trajectory is equal to the factor of all the probabilities of the segments it is composed of.

3.5.2 Trajectories

A *trajectory* is a possible path an object has followed in the scene. Each trajectory is composed of a set of linked segments that follow one another and a probability that defines how likely it is that the object tracked has followed the studied trajectory. The probability $P_o^{traj}(j)$ of a trajectory j of object o, composed of $N_S^{traj}(o,j)$ segments, is set following:

$$P_o^{traj}(j) = \min\left(P(o,s_i,t), D\right) \quad \forall 1 \leq i \leq N_S^{traj}(o,j) \tag{3.7}$$

where D equals $P^{dis}(o,t)$ if the trajectory ends at frame t, and $P^{dis}(o,t) > 0$; otherwise it equals 1. Based on the already known example, the trajectories of object $O1$ at time t_3 are shown in Figure 3.14

Figure 3.15 shows a second, more complex example. The objects leave the scene together after t_5 and the tracking system is unable to know the real trajectory each object has taken.

The probabilities of the top object ($O1$) being present in each segment are presented in Figure 3.16. Figures 3.17 to 3.19 illustrate the trajectories obtained. As is natural, the sum of the probabilities of all the possible trajectories of a single object is 1.

Usually, if new objects appear where another object was, due to a *case 5*, it means that the new objects could have been occluded under the previously present object, and so the new objects may have been in the same places at the same times as the other object. This is important information that the behavior stage of any security system has to take into account. As there is no *a priori* information about entrances and exits in the scene, the model cannot know if the object has really been occluded or has just appeared there. However, if the SDGM background subtraction algorithm (see Section 2.5) is used, this information is already available to the surveillance system. The newly visible object has to inherit the information about where the previous object has

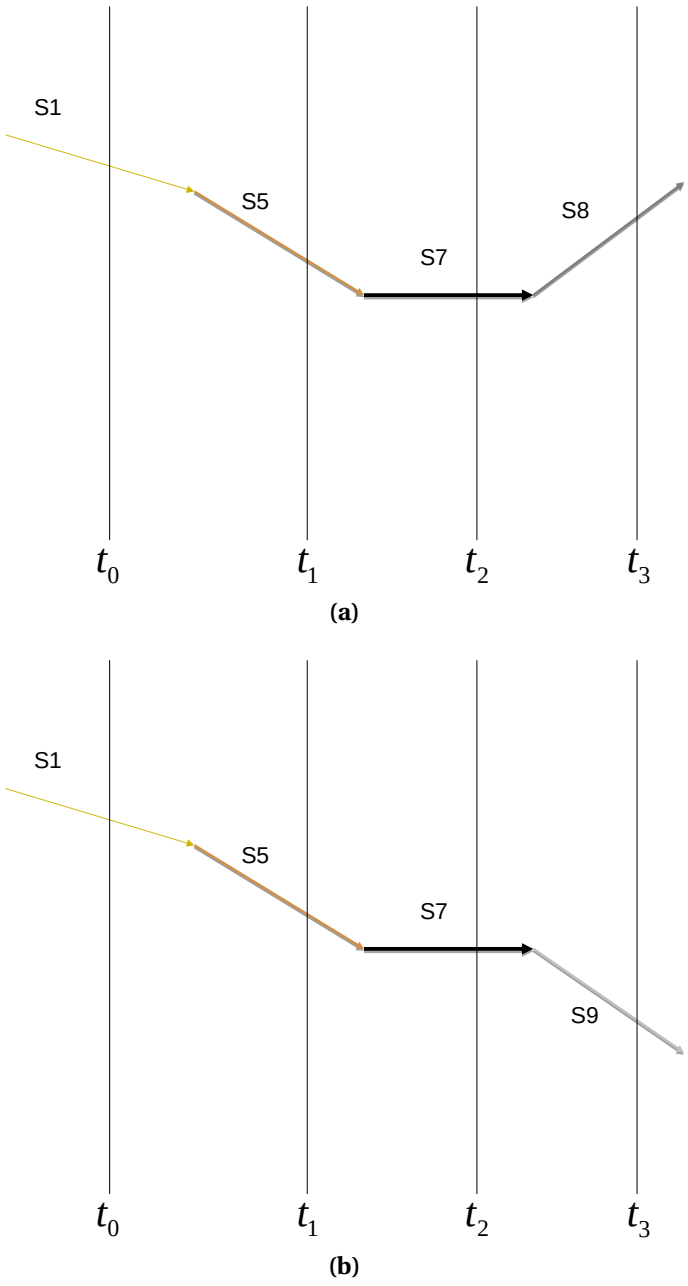


Figure 3.14. Examples of trajectories obtained at time t_3 for object $O1$. The probability of both trajectories is 0.5.

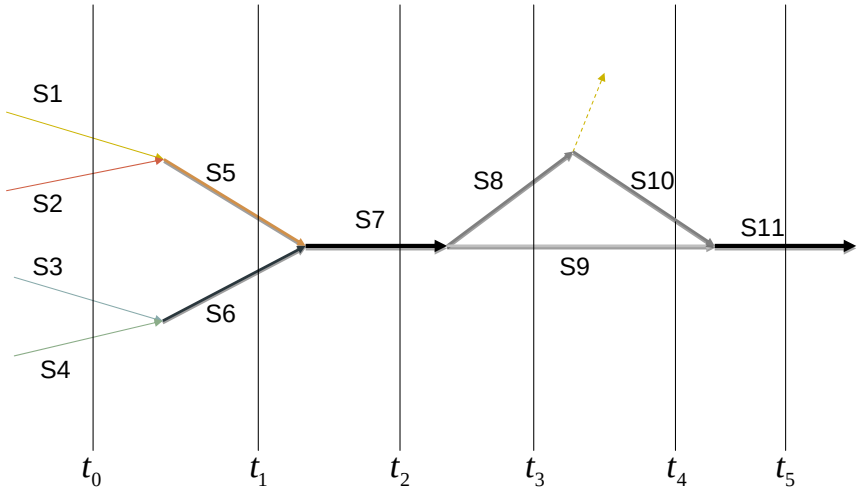


Figure 3.15. Second example of segments obtained after a complicated object interaction. The dotted line represents the probability of the object leaving the scene due to an exit at that place in the image, between t_3 and t_4. As no modification in $S9$ occurs between t_3 and t_4, the segment continues the whole time, while $S8$ may lead to an end or to the creation of a new segment $S10$ to continue the movement.

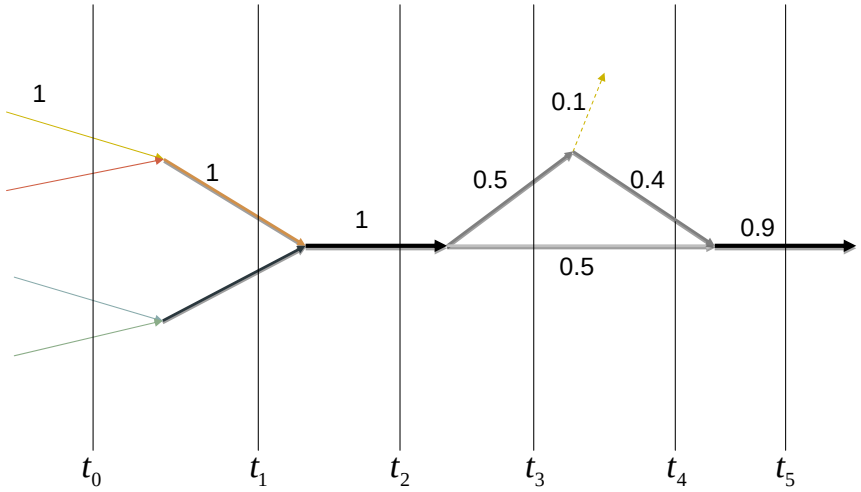


Figure 3.16. Probabilities of object $O1$ (top object) being present in each segment.

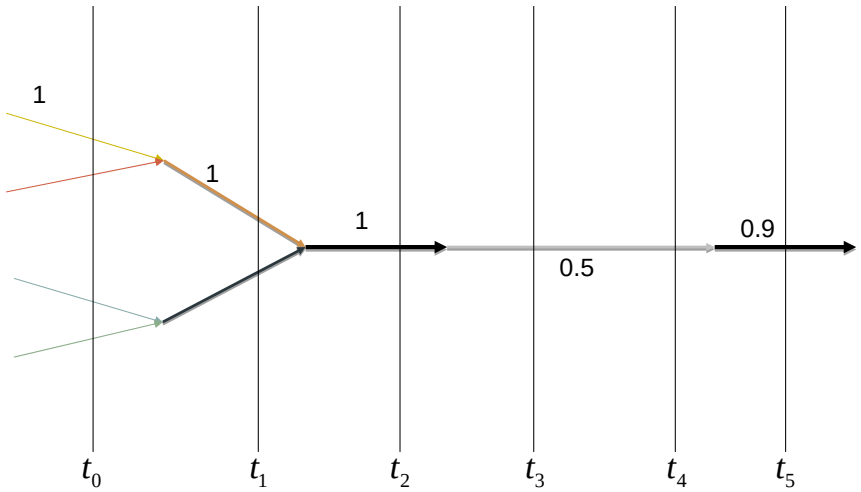


Figure 3.17. First $O1$ trajectory of second example, with a probability 0.5.

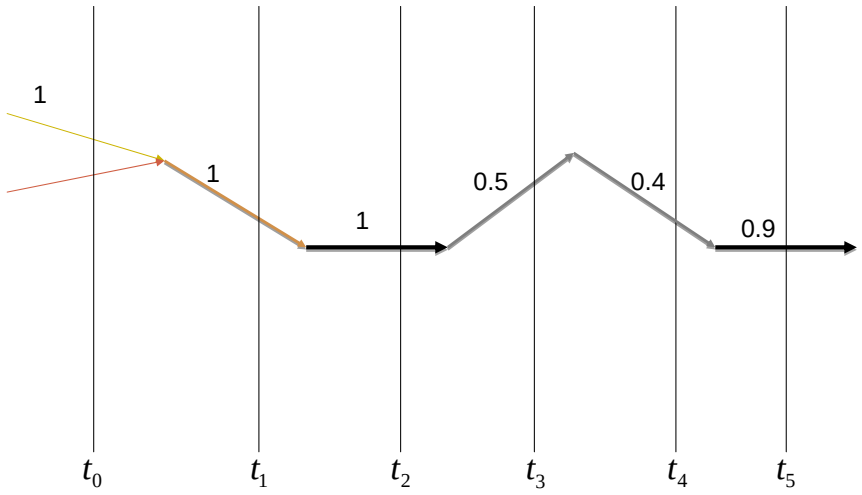


Figure 3.18. Second $O1$ trajectory of second example, with a probability 0.4.

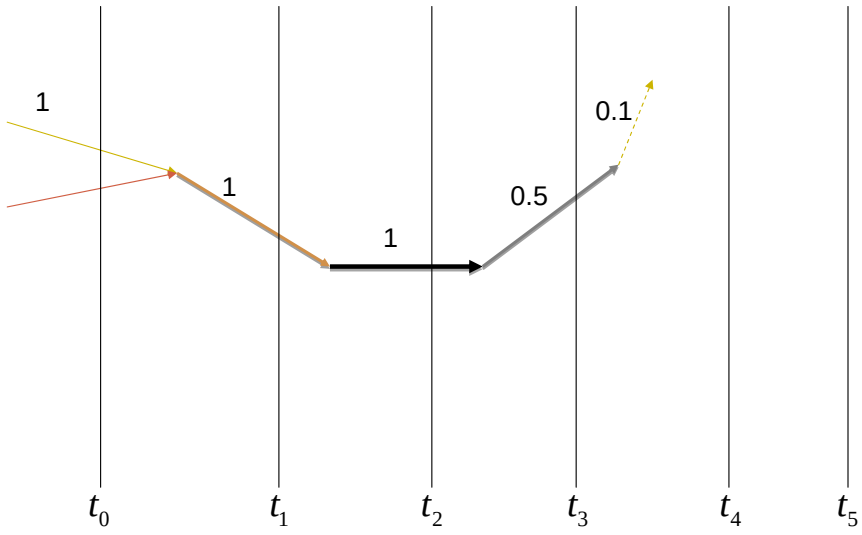


Figure 3.19. Third $O1$ trajectory of second example, with a probability 0.1.

been if the new object is considered to have been occluded under the old. Depending on the certitude of this assumption, a probability $P^{inh}(O_n, O_o)$ of object O_n inheriting trajectories from O_o is used. In Figure 3.9, $O14$ (man) and $O15$ (bag) have been occluded under the bigger object $O0$ (man and bag), thus both of them inherit the trajectories of $O0$, that is, they were present in any segment where $O0$ was present.

A new object O_n inheriting the trajectories of an old object O_o with a probability $P^{inh}(O_n, O_o)$ will be considered to be occluded in every segment where O_o has been present. The probability $P(O_n, s, t)$ of O_n being in segment s at time t where O_o has been with a probability $P(O_o, s, t)$ is:

$$P(O_n, s, t) = P(O_o, s, t) \cdot P^{inh}(O_n, O_o) \,. \tag{3.8}$$

Of course, if the probability of an object entering at the same location and at the same time that a tracked object splits in two (*case 5*) is set low enough, $P^{inh}(O_n, O_o) \approx 1$ for any O_n and O_o. Equation (3.8) is then simplified to:

$$P(O_n, s, t) = P(O_o, s, t) \,. \tag{3.9}$$

3.6 Results

In the following test, the probability of an object being in newly created segments was the probability of being in previous segment evenly split between the new segments. For example, in a *case 5* where there was a single object with probability 1 in previous frame and appears two blobs in present frame, the probability of the object being in any of the two new segmetns is 0.5. Parameter T_d is set to 0.35, having eight bins color histograms and five bins grayscale histograms.

A widely known set of videos has been used to test the presented tracking model. Goyette et al. present in [Goyette et al., 2012] a thorough video data set, also used in Chapter 2, divided in 6 different categories: *Baseline*, *Dynamic Background*, *Camera Jitter*, *Shadows*, *Intermittent Object Motion* and *Thermal*. These videos show several different challenging situations to determine the accuracy of a model. First, this section evaluates the global performance of the model. Secondly, some clarifying examples are shown to analyze special events in the videos.

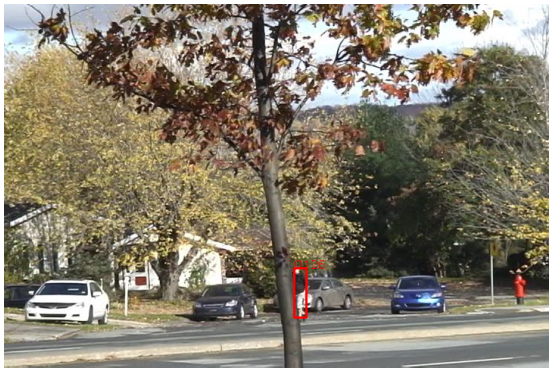
3.6.1 Global performance evaluation

Table 3.1 details the total number of occurrences of each case as well as the total number of detected trajectories per video.

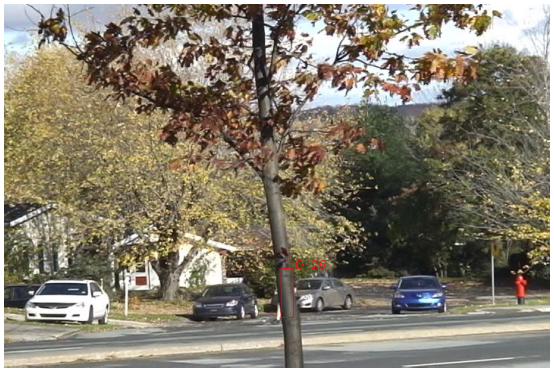
Table 3.1. Number of trajectories and occurrences of each case per video

Categories	Video	Case						Traject.
		1	**2**	**3**	**4**	**5**	**6**	
Baseline	Highway	6	67	708	6	5	29	**27**
	Office	1	15	1448	0	0	0	**1**
	Pedestrians	8	200	739	10	8	3	**9**
	PETS2006	8	120	1894	7	7	24	**9**
Camera Jitter	Badminton	6	53	530	7	7	72	**8**
	Boulevard	17	338	1134	2	3	1	**17**
	Sidewalk	8	152	467	0	1	0	**8**
	Traffic	18	326	493	0	1	22	**18**
Dynamic Backg.	Boats	2	22	1165	0	0	0	**2**
	Canoe	1	20	235	0	0	0	**1**
	Fall	27	422	1142	4	11	51	**31**
	Fountain01	8	99	198	4	7	0	**11**
	Fountain02	4	67	220	5	8	0	**7**
	Overpass	2	22	465	0	1	0	**2**
Intermittent Obj. Motion	SbandonedBox	8	117	2350	17	19	8	**8**
	Parking	3	60	597	0	0	0	**3**
	Sofa	7	108	4015	7	9	347	**13**
	StreetLight	4	45	2408	3	5	0	**3**
	WinterDriveway	3	33	1111	0	1	0	**2**
Shadow	Backdoor	3	55	654	1	1	0	**3**
	Bungalows	15	376	548	6	2	3	**19**
	BusStation	2	47	920	2	2	39	**2**
	CopyMachine	8	141	3059	8	10	42	**10**
	Cubicle	15	313	3083	3	3	4	**15**
	PeopleInShade	7	140	600	3	3	1	**8**
Thermal	Corridor	9	177	2845	2	2	0	**10**
	DinningRoom	3	29	3846	12	12	190	**12**
	LakeSide	6	197	4555	17	17	49	**7**
	Library	2	20	3628	0	1	0	**2**
	Park	6	118	311	4	4	0	**7**

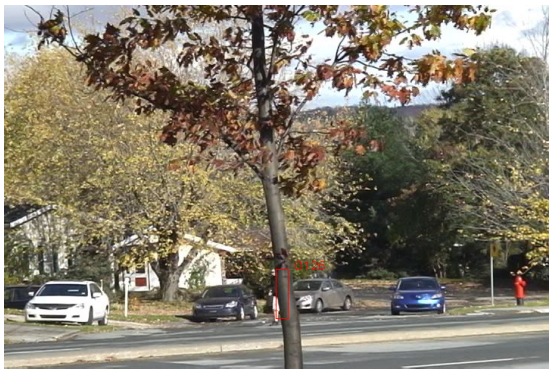
It can be observed that the number of occurrences of *case 3* is, obviously, considerably higher than the others, as it is the most frequent and usual case in any situation. Furthermore, there are several times more occurrences of *case 2* than of *case 1* (see Section 3.4). This does not mean that there have been many more objects deleted than created. When an object is no longer detected, that is, when a *case 2* occurs, it is maintained in the scene as occluded for several frames in order to be assigned to a new blob once the occlusion, if any, has finished. In each one of these frames, a new *case 2* is counted and thus the total number of *case 2s* is much bigger than *case 1s*. An example of the usefulness of such behavior is shown in Figure 3.20, where thick rectangles correspond to detected objects, and thin rectangles to the calculated positions of missing objects.



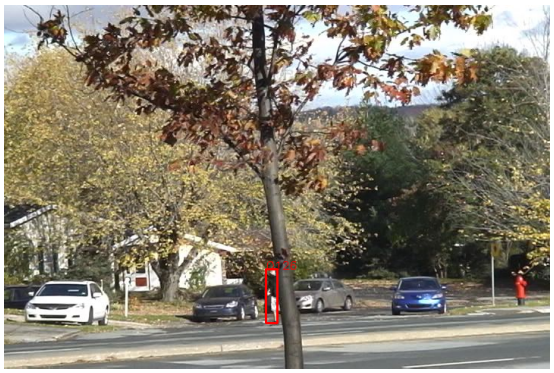
(a) frame 3016



(b) frame 3027



(c) frame 3031

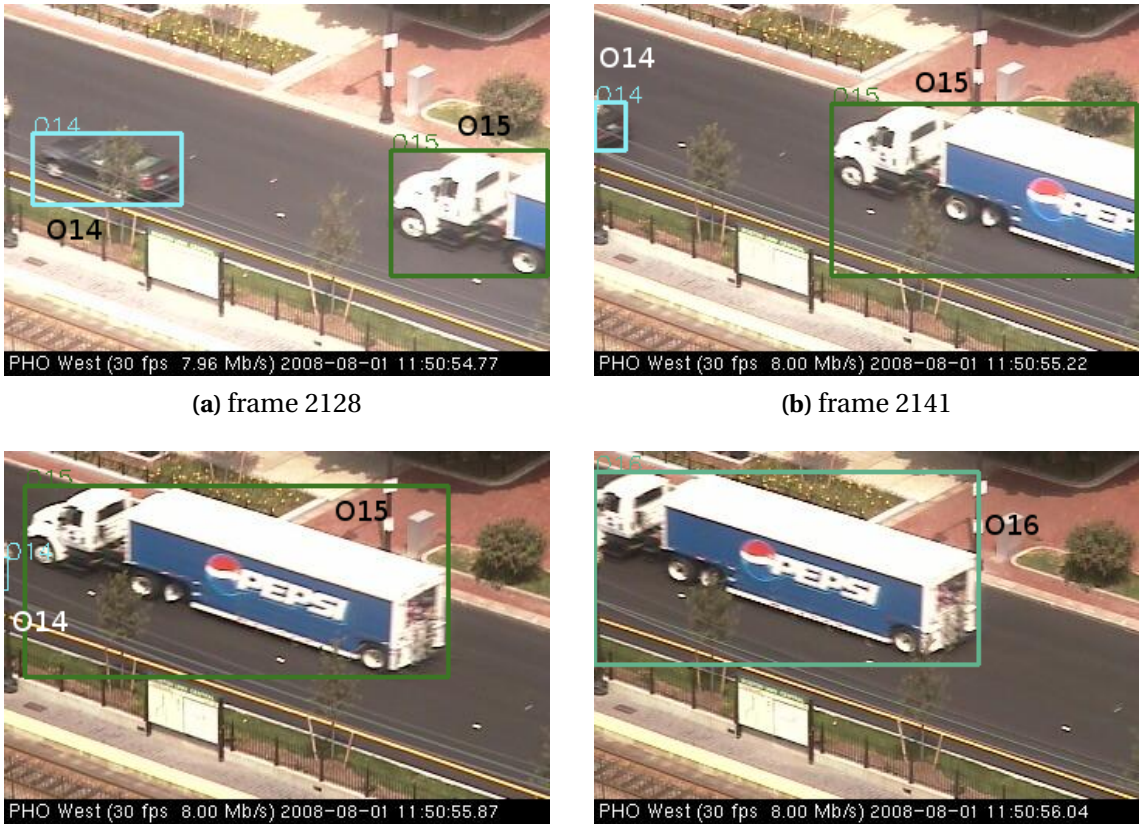


(d) frame 3033

Figure 3.20. Example of an occlusion in Fall video frames.

However, keeping a missing object in the scene may obviously lead to erroneously linking it with an existing object, as in Figure 3.21. The kept missing object, which in reality has left the image, is assumed to be occluded under what is actually a present object that moves near the disappearing point. To solve this problem, it is recommended to truly delete objects near the border of the image that have disappeared.

The errors made by the model are listed in Table 3.2. As *case 1* and *case 2* problems are intimately related, the wrongly created/finished tracks are grouped in the third column. These errors occur when the model considers a track to be finished and then



(a) frame 2128 **(b)** frame 2141

(c) frame 2160 **(d)** frame 2165

Figure 3.21. Example of false occlusion leading to an erroneous trajectory in Boulevard video. Object O16 contains both O14 and O15. The thin rectangle means that an object is not detected but may be occluded.

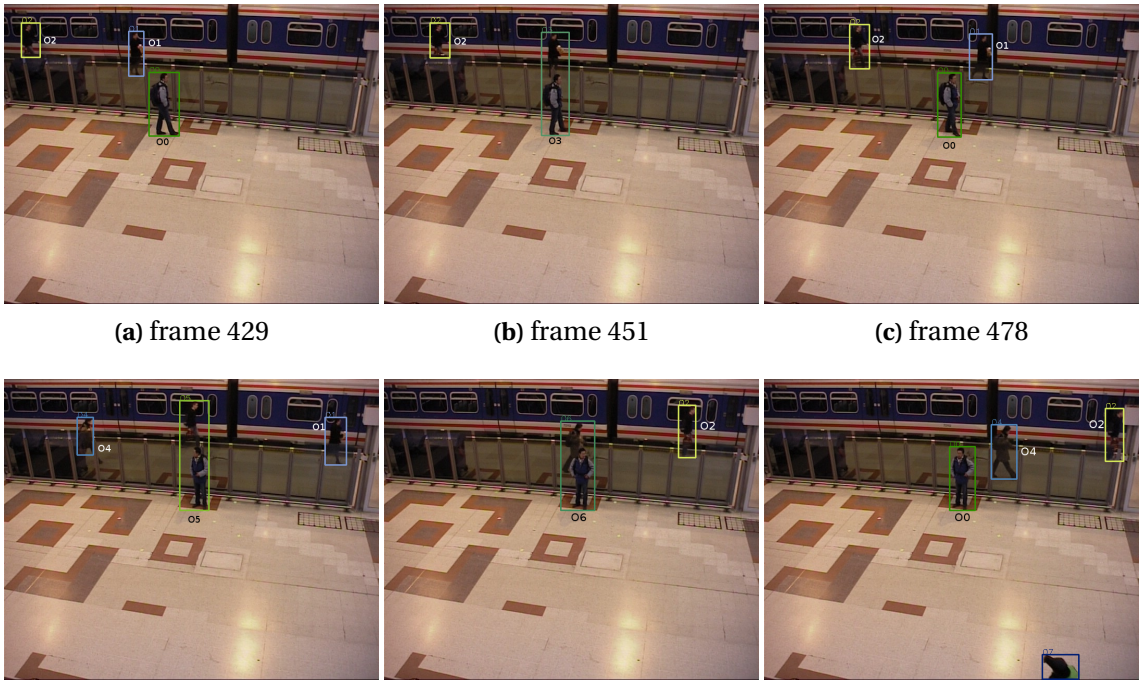
creates a new track, while it is the same person that has momentarily left the scene. As stated before, *case 3* corresponds to the most usual case and it does not bring any difficulty to a tracking model, so it has been removed from the table as there has been no error in any of the videos. Finally, *case 4* corresponds to the cases where there were several objects in the scene in the previous frame and only one has been detected in the present frame. The objects are then occluding each other. A perfect model would distinguish and tell them apart as a human being could. In a way, it could be said that each *case 4* is an error, however, given the current state of the art it is very costly in time and resources to to make such a distinction, which would also provide far from perfect results, so it is common to consider the objects occluded by others. Current tracking algorithms, like this one, focus on solving the problem of differentiating the objects once separated (*case 5s* and *case 6s*). Thus, *case 4* is also left out of the table, assuming *case 5s* and *case 6s* carry the weight of the model in the difficult scenes.

Table 3.2. Number of errors per video

Categories	**Video**	**Case**			**W. Traj.**
		1 - 2	**5**	**6**	
Baseline	Highway	0	0	3	**0**
	Office	0	0	0	**0**
	Pedestrians	0	3	3	**0**
	PETS2006	1	0	0	**0**
Camera Jitter	Badminton	4	2	31	**4**
	Boulevard	0	0	0	**0**
	Sidewalk	1	0	0	**0**
	Traffic	0	0	0	**0**
Dynamic Backg.	Boats	0	0	0	**0**
	Canoe	0	0	0	**0**
	Fall	14	2	45	**4**
	Fountain01	8	1	0	**0**
	Fountain02	2	1	3	**1**
	Overpass	0	0	0	**0**
Intermittent Obj. Motion	AbandonedBox	2	2	4	**2**
	Parking	0	0	0	**0**
	Sofa	0	2	74	**12**
	StreetLight	0	2	0	**0**
	WinterDriveway	1	0	0	**0**
Shadow	Backdoor	1	0	0	**1**
	Bungalows	0	5	0	**0**
	BusStation	0	0	2	**1**
	CopyMachine	1	1	1	**3**
	Cubicle	2	0	1	**0**
	PeopleInShade	0	1	0	**2**
Thermal	Corridor	2	2	0	**1**
	DinningRoom	1	10	171	**12**
	LakeSide	2	1	11	**2**
	Library	0	0	0	**0**
	Park	0	3	0	**3**

3.6.2 Examples of special events

This subsection presents in detail several examples of incorrectly tracked objects to try to explain the small flaws this model may have. As can be seen, on several occasions the model perfectly tackles all the different cases and perfectly tracks all the trajectories. Some examples of correct tracking are shown in Figures 3.22 and 3.23.



(a) frame 429 **(b)** frame 451 **(c)** frame 478

(d) frame 536 **(e)** frame 599 **(f)** frame 623

Figure 3.22. Example of a correct tracking in the PETS video. Objects O3, O5 and O6 contain two objects.

The presented model detects more trajectories than people in the scene. It can model more than one trajectory for a person, taking, for example, the torso and the head of a person as two different interesting objects (see Figure 3.24). If compared to the ground truth, there are more trajectories than expected, but all of them model real behaviors in the scene and are thus correctly formed to be an acceptable input to the behavior analysis module (see Chapter 4). Moreover, although there are some errors in certain cases in some videos, these errors are corrected in the following frames thanks to the new information taken into account in subsequent frames, as seen in Section 3.5. An example of this behavior can be seen in Figures 3.25 and 3.26.

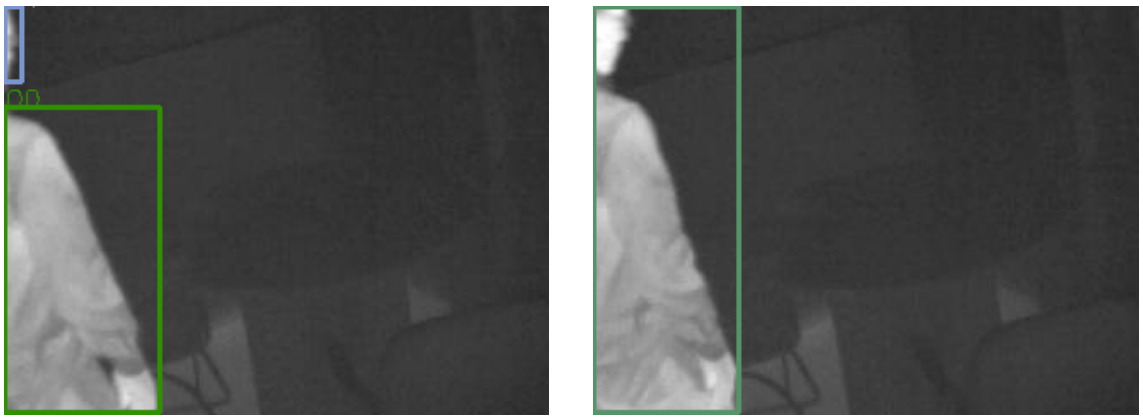
However, the model incurs in several small errors. If a new object appears in the same zone where another is leaving, the model may identify them as a unique object that has changed course, as in Figure 3.27. The model should have deleted the old object and created a new one, obtaining two trajectories instead of one erroneous trajectory.

On rare occasions, there are new trajectories created unnecessarily (see Figure 3.28), computed as error in *case 1* in Table 3.2. It can be seen that object $O17$ will correctly carry

(a) frame 5510 **(b)** frame 5564 **(c)** frame 5581

(d) frame 6500 **(e)** frame 6574 **(f)** frame 6620

Figure 3.23. Example of a correct tracking in the Cubicle video. Object O10 contains objects O8 and O9.

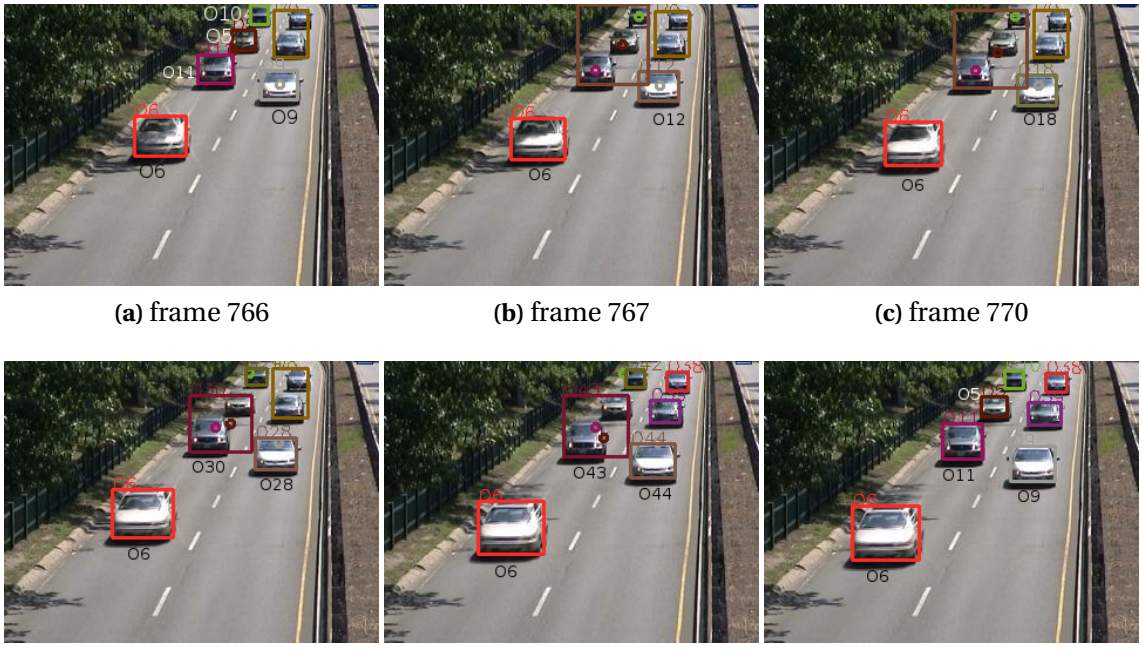


(a) frame 906 **(b)** frame 913

Figure 3.24. Example of a union in a unique object (teal) of two objects (green and blue) modeling limbs of a person in the Library thermal video.

all information of the trajectory, while object *O*18 will model an erroneous trajectory as it is not possible to enter the scene where it appeared. Of course, these problems are easily solved with a combination of SDGM (see Section 2.5) and tracking. Moreover, even if not solved, these errors do not disturb behavior analysis. The confusion of creating a new object where it is simply the continuation of another partially occluded object is done continuously, so the behavior analysis module is trained to understand that a trajectory beginning at that spot is a normal one, and no false alarm will be triggered.

Finally, a palette of colors that is too similar for different objects may mislead the tracking module. An obvious case can be found in thermal videos, as they do not contain



(a) frame 766 **(b)** frame 767 **(c)** frame 770

(d) frame 775 **(e)** frame 780 **(f)** frame 782

Figure 3.25. Example of a correction of previous error in the Highway video. Object O12 contains objects O5, O9, O10 and O11; object O18 contains O5, O9 and O10; object O30 contains O5 and O11 and objects O28 and O44 contain O9 and O11.

color information. Figure 3.29 shows the resulting tracking and the histograms of such a situation.

It can be seen that the model struggles in two single videos, namely *Sofa* where people with a similar color palette occlude objects and move them around, and *DiningRoom* which is a thermal video (without color information) where the person moves around and occludes several objects as well. Overall, the number of errors while analyzing cases is below 1%.

3.7 Conclusion

Tracking interesting objects detected in a surveillance video camera is a challenging task as well as a necessary step to understand the behavior of these objects. From a binary image with pixels in motion, blobs are obtained and used to perform tracking. This chapter proposed a color space to avoid problems by generating color histograms with dark, bright, or gray pixels. Blobs and the resulting tracked objects are modeled by a pair of histograms composed of gray and color pixels respectively. The tracking between previously tracked objects and currently detected blobs is carried out by means of space proximity and color distance. Moreover, there is an explanation of the six simpler cases of the scenarios that a tracking model has to deal with. Finally, a probabilistic model to tackle occlusions is presented. The model can manage different possible trajectories

(a) frame 306

(b) frame 417

(c) frame 429

(d) frame 440

(e) frame 452

Figure 3.26. Example of a correction of previous error in the Pedestrians video. Objects O2 and O10 contain objects O0 and O1 and objects O4 and O9 contain objects O0, O1 and O3.

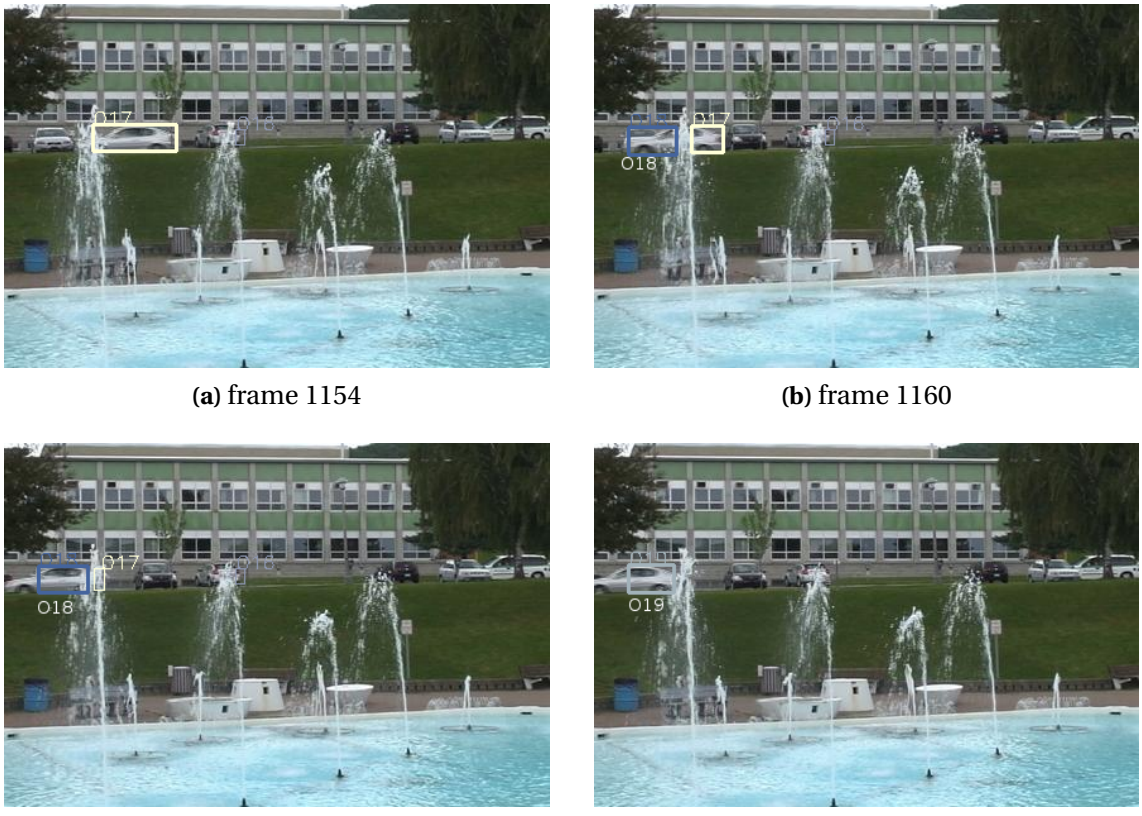
with different probabilities for a single object, as well as update probabilities taking into account the new information obtained in subsequent frames.

The model has been tested using common known dataset videos, where it has demonstrated its ability to track in several different situations. The overall error rate is kept low enough, under 1%, thus confirming it as a reliable tracking system to use.

This model provides the information necessary for a behavior analysis algorithm, an essential element of modern security applications.

(a) frame 1751 **(b)** frame 1775 **(c)** frame 1803

Figure 3.27. Example of an object entering while another exits in the Backdoor video.

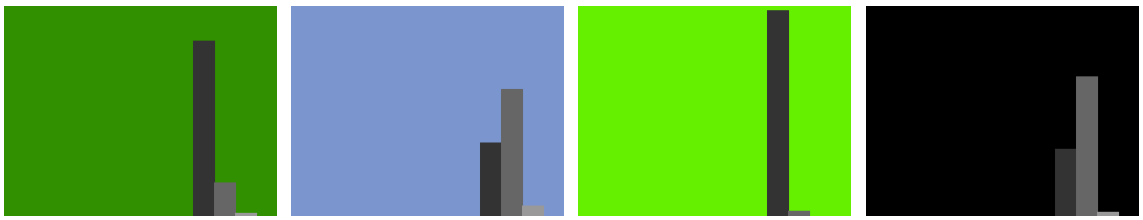


(a) frame 1154 **(b)** frame 1160

(c) frame 1164 **(d)** frame 1166

Figure 3.28. Example of incorrectly created object in the Fountain01 video. Object O19 contains both O17 and O18. The thin rectangle means that an object is not detected but may be occluded.

(a) frame 2166 **(b)** frame 2235 **(c)** frame 2411



(d) O0 **(e)** O1 **(f)** left blob **(g)** right blob

Figure 3.29. Several Lakeside thermal video frames, with the color histograms of the objects before merging (frame 2166) and the blobs once split (frame 2411).

Chapter 4

Behavior Analysis

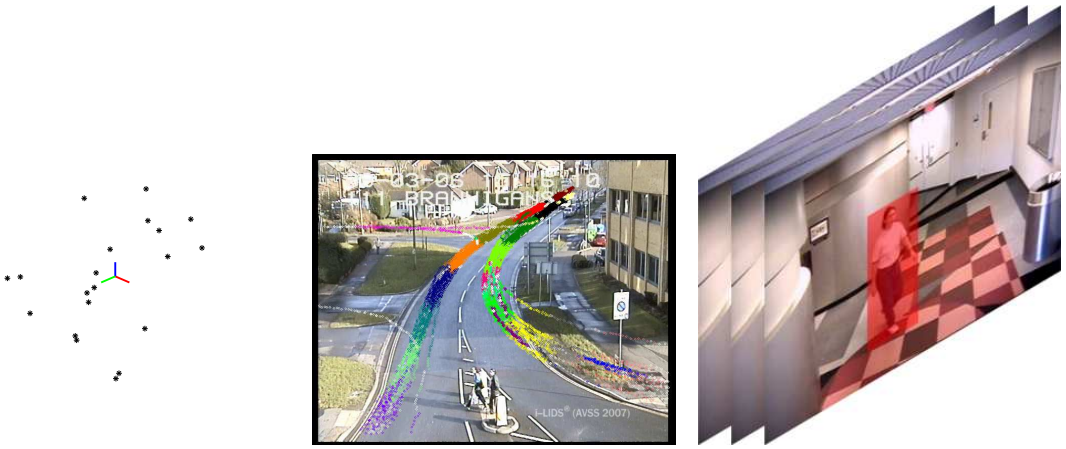
4.1 Introduction

The first two steps of surveilance systems, motion detection and object tracking, analyzed in Chapters 2 and 3 correspond to lower level image processing techniques, while behavior analysis is solved at a higher level through machine intelligence techniques. Although image processing is a widely developed area, relatively little research has been carried out in understanding human activities [Jan, 2004, Ko, 2008].

Nayak et al. [Nayak et al., 2011] point out that one important task of surveillance systems is to detect suspicious or anomalous activities. In many cases, it is too difficult to explicitly define what all the abnormal behaviors are. Furthermore, the system has to even handle events that have not previously occurred. This is why these systems usually only learn normal behaviors and consider any unknown behavior as an abnormal behavior [Jiang et al., 2011].

Figure 4.1 shows the main steps in an abnormal behaviors detection model. The first thing is to select the components to be used. In the literature, position, speed and direction are the usual components employed (see [Hu et al., 2006, Jiang et al., 2011]). Once chosen, the model has to be trained. Generally, the movements observed are partitioned into discrete states and then classified [Ko, 2008]. If a training set is available, Hidden Markov Models can be used, such as the Coupled Hidden Markov Models [Brand et al., 1997, Oliver et al., 2000]. When there is no dataset available to do the training of the system, an unsupervised clustering is usually performed, where dense clusters correspond to normal behaviors whereas rare events are considered anomalous [Nayak et al., 2011]. Finally, the model is confronted with real data where it has to be able to discern between normal and abnormal behaviors.

In this chapter, the Dominant Sets framework is used to detect abnormal behavior in a tracking scenario (see [Alvar et al., 2014b]) , where the focus is set only to detect deviations of the tracks from the normal motion patterns observed, and not to detect



(a) Component selection **(b)** Model training **(c)** Anomaly detection

Figure 4.1. Behavior model steps to detect anomalies.

appearance anomalies like, for example, a car moving in a pedestrian area. To this end, the problem is cast as one of outlier detection in the tracks, following an approach similar to other clustering-based approaches in the literature. In a departure from previous attempts with the framework, however, the approach is not used to directly extract a meaningful normal behavior class, contrarily to semantic interpretation techniques. Rather, the approach is allowed to over-segment the behavior space, thus using the approach as a robust non-parametric density estimation approach. This abnormal behavior detection model presents several contributions. First, an implementation of Dominant Sets for the detection of quick abnormal behaviors is detailed. Secondly, this model's capability for use in any security system and in any environment is examined. This capability is due to the fact that it does not use any application-specific knowledge or any additional information from the scene. Moreover, no labeled training data set is necessary, as the normal behavior is learned in an unsupervised way. Finally, unlike other clustering methods, it does not need to initialize the number of acceptable normal behaviors.

The structure of this chapter is as follows. The following section contains comments on the state of the art. The Dominant Sets method is explained in detail in Section 4.3. Next, Section 4.4 details representation of the behavior using Dominant Sets, followed by Section 4.5 that shows the results obtained with the model here presented. Finally, the conclusions are drawn in the last section.

4.2 State of the art

Behavior understanding consists in the recognition and description of the different activities carried out by the interesting person followed. This step of a surveillance system tries to match a previously learned pattern of activities with the present activities

observed in the scene.

4.2.1 Semantic interpretation

In many applications, it is essential to describe the behavior of the studied object in natural language. The conceptual descriptions of the actions in the scene are the best mechanism to communicate what is happening in the image to humans [Buxton and Mukerjee, 2000]. That is why semantic interpretation has become an attractive and active topic [Lou et al., 2002, Wang et al., 2003].

The use of models such as Bayesian networks allows the interpretation of some events and behaviors. These methods stay in a low level framework and are usually not capable of explaining complex behaviors, which would involve higher level reasoning. Remagnino et al. [Remagnino et al., 1998a, Remagnino et al., 1998b] propose a method to obtain a semantic interpretation of the scene. First, they use Bayesian networks to recognize human behaviors: their speed, acceleration, direction and movement curvature. Then, once this information has been obtained from different people, in addition to the location of interesting objects, the method can describe the relative location with respect to each other - close, approaching, getting away, etc.

These methods utilize symbol systems to represent behavior patterns. Kollnig et al. [Kollnig et al., 1994] analyze traffic videos using fuzzy membership functions to match verbs with attributes obtained in the tracking stage. Each occurrence is defined in three steps: a precondition to trigger the event, a monotonicity condition indicating the direction and a postcondition that models the end of the event.

Kojima et al. [Kojima et al., 2000] present a new method for analyzing human behavior. First, the head region of the person is detected. The results from a dimmed edge extractor are compared to samples of known poses to understand the actual pose of the person and where he is facing. Secondly, the size and location of the head in the image allows the program to locate it in the scene, using previously calculated projections of the room and knowledge of its structure. Then, some features - proximity to regions of interest or other objects, changes of direction, etc.- are extracted. A series of certain behaviors lead to the conceptualization through verbs and finally to a whole simple sentence that follows the structure: "AGENT ACTION-VERB OBJECT [WHEN]". In [Kojima et al., 2002], they expand the results taking into account the position and movements of hands, allowing the recognition of new events as taking/leaving objects.

In [Lou et al., 2002], the trajectories of vehicles are used to understand vehicle behavior. Initially, all training trajectories are compared to separate them into clusters. The comparison is made using the Hausdorff distance and the C-Mean clusterization method. Thereupon, the trajectories are divided in sectors, each of which is associated with an action: move forward, turn left, turn right and stop. Training leads to a classification tree that helps to predict the next movements of vehicles. Each new

detection updates the tree. Lastly, the trajectory data, added to the knowledge of the environment, allows the system to generate a description of the events in the scene with natural language, using sentences similar to "AGENT ACTION-VERB WHERE [at low/middle/high speed]".

The main advantage of semantic interpretation techniques is that the behavior analysis is complete and fully understandable by a non-experienced operator. However, these techniques require a costly and extensive supervised training, rarely available in newly installed systems.

4.2.2 General Techniques

Dynamic Time Warping (DTW) is a technique based on patterns originally used in voice recognition (see [Myers et al., 1980]). An example of the use of this technique in the field of behavior analysis can be seen in [Bobick and Wilson, 1997] where the authors train the system to recognize certain patterns of movement. The image is divided into zones, and each zone is considered to model a state. When an interesting object in movement passes through a zone, it is considered to be in the associated state. A series of states defines a behavior. A good point of this technique is that it is able to match a movement to a pattern independently of the time. The pattern and the movement analyzed can have different speed, the only important factors are the temporal location and the order of the states. Patterns are also used by Chomat and Crowley, who generate movement templates by applying PCA to a set of spatio-temporal filters [Chomat and Crowley, 1998]. These filters are generated from a set of image sequences. The selection of components is performed thanks to Bayes' rules. More recently, Bobick et al. presented another algorithm based on templates used for recognizing behaviors [Bobick and Davis, 2001]. In this case, the movement is stored for several frames, in order to obtain a binary black and white image with zones in movement, and a second similar image in grayscale where the clarity of each pixel indicates the proximity in time of the movement. The matching with known templates is performed using the Mahalanobis distance. The main drawback of any template-based technique is the need to create and classify the templates. That is, a set with supervised *normal behavior* templates should be delivered, which is not always available.

State machine techniques are used to determine whether any reference sequence of images matches with the analyzed sequence or not. Wilson et al. [Wilson et al., 1997] use a finite state machine with only 3 states - rest, transition, stroke - to model the behavior of a person telling a story. A sequence has to fulfill a set of constraints such as state duration and minimum movement speed to be considered as of one state or another. On their part, Brémond and Medioni create a state machine to recognize scenarios [Brémond and Medioni, 1998]. They used the method to recognize vehicle behaviors registered by an airborne video camera. Each scenario is composed of several subscenarios, each one corresponding to different states - such as "accelerating" or

approaching a "checkpoint". They assert that this technique can be used to identify human behaviors as well. The state machine techniques are suited to knowing which action, of the different actions known to the program, the people are doing. Thus, similarly to other semantic interpretation techniques already seen, it requires a huge amount of supervised data to model the algorithm correctly and is not adapted to recognize new actions or states.

Hidden Markov models (HMM) (see [Poritz, 1988]) are a technique that requires two stages: training and classification. In general this technique outperforms other template techniques such as DTW [Wang et al., 2003, Hu et al., 2004a]. Bregler [Bregler, 1997], analyzes body movements once the image has been segmented and the different parts of the body have been tracked. He then uses HMM to deduce human behaviors, that is, complex movement, thanks to the simple limb movements. Some authors, as in [Brand et al., 1997], claim that Coupled Hidden Markov models have even better results than HMM. The same research group uses both methods to model human behaviors and interactions such as meetings or one person following another [Oliver et al., 2000]. Unfortunately, supervised training sets are not always available and they restrict the applications of the behavior detection model considerably.

The model presented in [Adam et al., 2008] needs minimal setup and allows low computational cost for unusual event detection. Low-level feature observations such as optical flow patterns are used to discriminate normal and abnormal behaviors. Nevertheless, this model cannot detect unusual events characterized by several normal actions. Antic and Ommer [Antic and Ommer, 2011] use templates of the training data combined with the optical flow to set up a probabilistic model to indirectly detect abnormalities. Li et al. [Li et al., 2010] compute the optical flow in foreground zones. Then, they apply a particle system [Ali and Shah, 2007] and its energy is used to decide if the trajectories analyzed correspond to abnormal behaviors. A similar approach for crowded scenes is presented in [Kaltsa et al., 2012] using optical flow and particle advection. Kim et al. model local optical flow patterns with a mixture of probabilistic PCA models and use Markov Random Fields (MRF) to ensure global consistency in [Kim and Grauman, 2009].

In [Cong et al., 2011, Cong et al., 2013], the authors present a model to detect abnormal behavior on a local and global basis as well. They use the sparse reconstruction cost over the normal dictionary to measure how normal a sample is. To limit the size of the dictionary, they also propose a new selection method. Other methods focus on the pixel-level features for being robust in crowded scenes (see [Gong et al., 2011]) and avoiding problems derived from the tracking step. Another method to perform behavior characterization based on spatial and temporal dependencies between motion labels obtained with background subtraction is presented in [Benezeth et al., 2009], where they only use background subtraction data, avoiding having to perform the tracking step. In [Zhao et al., 2011], the authors use crowd instability to describe abnormal crowd behaviors. First, the image is divided in blocks. Then, social force and optical flow are

calculated to evaluate the spatial instability in each block. Finally, a statistical analysis of block instability allows temporal instability to be determined. Mehran et al. also use social force in [Mehran et al., 2009], combined with Latent Dirichlet Allocation (LDA) to detect abnormal behaviors. Mahadevan et al. [Mahadevan et al., 2010, Li et al., 2014] propose using a mixture of dynamic textures (MDT) instead of optical flow to discriminate saliency scores spatially and temporally in different scales. They not only detect abnormal features, but also localize the abnormality. The main drawback of this technique is the computation time needed.

The common solution, when there is no dataset available to do the training of the system, is to use an unsupervised clustering, where dense clusters correspond to normal behaviors whereas rare events are considered anomalous [Nayak et al., 2011]. A fuzzy K-means-based algorithm is presented in [Hu et al., 2006]. The clustering takes place in a two-step model, where trajectories are first clustered on a spatial basis and then each cluster in turn is sub-clustered on a temporal basis. Both clusterizations are performed with a fuzzy K-means algorithm. The abnormal full and partial trajectory detection is carried out using Bayes' rules. Stauffer [Stauffer, 1999, Stauffer and Grimson, 2000] presents a hierarchical activity classifier. This method generates a codebook of prototypes and obtains a co-concurrence matrix over the prototypes. A binary tree representation of the different behaviors is constructed from this matrix. Although Stauffer claims that an unusual event detector may be built from this work, the details of how to obtain this detector are not provided. Based on [Benezeth et al., 2009], Saligrama and Chen introduce not only the background subtraction as feature, but also the optical flow of pixels in each spatio-temporal mask [Saligrama and Chen, 2012]. They obtain an 11-dimensional descriptor that, when combined with a trained Nearest Neighbors clustering algorithm, detects local anomalies. Neural networks are used as clusterization methods to know the behavior in several applications, such as using time-delay neural networks for lip reading [Lin et al., 1999, Meier et al., 2000] or hand gesture recognition [Yang and Ahuja, 1998]. Guo et al. [Guo et al., 1994a, Guo et al., 1994b] model each tracked person as a stick figure to analyze people's behavior in a scene. They then use the first four Fourier transform components of a person's movement as the input values of the neural network. Johnson et al. [Johnson and Hogg, 1996] propose a neural network for atypical instantaneous movement and trajectory detection. They define flow vectors as the input vectors composed of the location and speed of the object. This method has been updated and improved by several researchers [Sumpter and Bulpitt, 2000, Owens and Hunter, 2000]. [Hu et al., 2004b] present a neural network where the input values are the location and speed of the tracked object in every frame as well as its size, and the number of output neurons corresponds to the number of behavior clusters. Trajectories are "normalized" to the same length, adding new points to smaller trajectories. The Euclidean distance from each neuron to new trajectories is calculated. If it exceeds a threshold, the new trajectory is considered abnormal, and then a point-to-point distance is calculated to find out exactly where the anomaly has taken place.

A drawback of these works is that the number of clusters/prototypes/neurons is not estimated and has to be set manually.

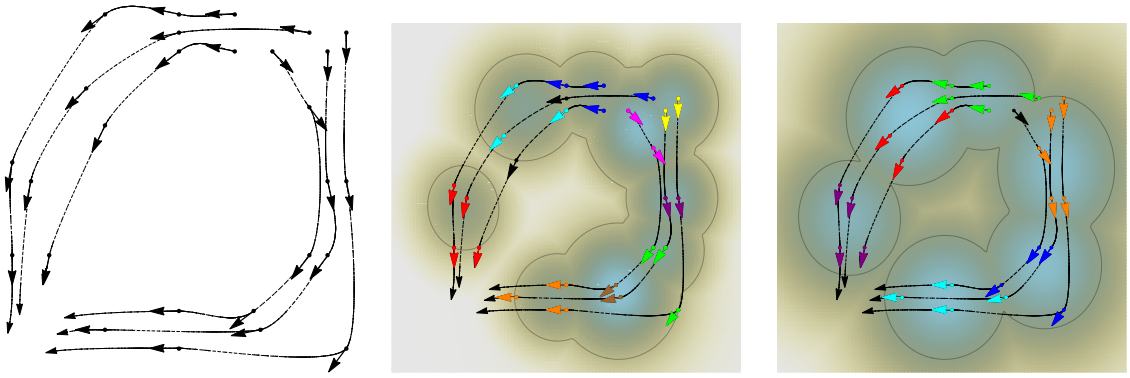
Dee and Hogg [Dee and Hogg, 2004a] propose an approach based upon building an understanding of the tracked people's goals. The aim is to detect interesting behaviors [Dee and Hogg, 2004b] analyzing whether the trajectory to reach a goal (exit) has been the logical one. This method allows an unknown behavior to be labeled as normal, which is impossible for usual methods, but needs a huge amount of manually generated data. Xiang et al. [Xiang and Gong, 2005, Xiang and Gong, 2008] propose a novel clustering framework that does not need to know the number of clusters *a priori* and uses Dynamic Bayesian Networks (DBN) to measure similarity, a spectral clustering based on eigenvectors and a Likelihood Radio Test (LRT) method to decide if a behavior is normal or not. Unfortunately, although it is fast and without manual labeling of training sets, their framework needs a large number of thresholds, which minimizes the gain obtained from manually setting the number of clusters.

Dominant Sets [Pavan and Pelillo, 2007] and its game theoretic generalization [Torsello et al., 2006], is an unsupervised learning framework that works in an unconstrained similarity setting. One key characteristic of the approach is that it departs from the data partition paradigm that dominates clustering approaches to concentrate on the key characteristics of a cluster [Pelillo, 2009], namely the cluster's internal homogeneity and its distinctiveness with respect to external elements. In a sense, the approach is more closely related to one-class classifiers than to clustering approaches. Furthermore, the approach has shown to be particularly selective, and for this reason it has been used effectively in contexts such as inlier selection [Albarelli et al., 2009, Albarelli et al., 2012], feature selection [Albarelli et al., 2010, Zhang and Hancock, 2011], prototype selection [Vascon et al., 2013] and robust neighborhood construction [Yang and Latecki, 2011]. The selectiveness and one-class characteristic of the framework render it particularly useful in behavior analysis, where typically the various behaviors are asymmetrically covered. Wei et al. [Wei et al., 2007] use Dominant Sets to cluster detected actions into classes and use the out-of-sample approach presented in [Pavan and Pelillo, 2004] to generalize the learned classes. Hung and Kröse [Hung and Kröse, 2011], on the other hand, use Dominant Sets to detect specific dispositions of crowds called F-formations. Hamid and co-workers [Hamid et al., 2005, Hamid et al., 2009] have used the Dominant Sets framework to detect abnormal human behavior. They cluster action n-grams, i.e., fixed short sequences of atomic actions, and then detect abnormal behavior as actions not belonging to any cluster.

As stated, the Dominant Sets approach is not used to directly extract meaningful normal behavior class, but to over-segment the behavior space. Dominant Sets is used as a robust non-parametric density estimation approach. Abnormal behavior is then defined as one in a low-density area of the phase space. This allows the approach to work reliably even in low-level motion features, as opposed to the high-level actions n-grams adopted in [Hamid et al., 2005, Hamid et al., 2009]. In this context, the main advantage

of using dominant sets over partitional approaches, such as k-means clustering, or density estimation approaches, such as mixture of Gaussian or kernel density estimators, is the fact that dominant sets implicitly provides a decision boundary against the null hypothesis, i.e., a boundary against unseen classes, whereas partitional-based approaches can only provide boundaries between already seen classes, and density estimation approaches require a parameter to be tuned specifying the size of the tail to be cut. This property is particularly important for abnormal behavior detection where the system can be trained only with normal samples.

While any other one-class classifier, such as the one-class SVM [Schölkopf et al., 1999], would still work within this context, Dominant sets approach has been chosen because it is just as efficient, it provides an equally sparse representation (the clusters), and the decision boundary is locally adaptive, as it depends on the size and spread of the closest cluster. In contrast, in one-class SVM, the boundary is determined by a fixed global margin in the implicit feature space.



(a) Original normal data **(b)** Clusters and decision boundary ($c = 1$) **(c)** Density and decision boundary ($c = 5$)

Figure 4.2. Extraction example of normal/abnormal class boundary using dominant sets. The arrows represent position and velocity of tracking observations along six trajectories (dotted lines). The decision boundary for normal data is obtained by clustering the observations and comparing the similarity of new data to the cluster elements with the average class similarity (see Equation (4.17)). As the decay of the data similarity (parameter c) changes, so does the spreading and number of clusters and the resulting area of normal behavior in feature space. The image shows the boundary for the best possible velocity, traces with unaligned velocities will have tighter boundaries.

Figure 4.2 shows a synthetic illustration of the overall Dominant Sets training process: first, the normal behavior data is over-segmented into several small clusters. The size of the clusters is based on a scale parameter c, which is the only parameter of the approach. Once the clusters are at hand, then the data density and decision boundary are obtained with Equation (4.17). Note how the size of the boundary obtained through the dominant sets framework adapts to the number and dispersion of the data in each sub-cluster.

4.3 Dominant sets

4.3.1 Dominant Sets extraction

Let the data to be clustered be represented by a set $V = \{1, \ldots, n\}$ of objects, with an $n \times n$ matrix $A = (a_{ij})$ representing the similarity between a pair of objects, where all elements on the main diagonal are zero. Further, let $S \subseteq V$, the *(average) weighted degree* of i with respect to S be defined as

$$\text{awdeg}_S(i) = \frac{1}{|S|} \sum_{j \in S} a_{ij} \tag{4.1}$$

where $|S|$ denotes the cardinality of S. Moreover, if $j \notin S$ the quantity

$$\phi_S(i, j) = a_{ij} - \text{awdeg}_S(i) \tag{4.2}$$

defines a measure of the similarity between nodes j and i, with respect to the average similarity between node i and its neighbors in S. Let $S \subseteq V$, $i \in S$ and $S \setminus \{i\}$ be the set S excluded i. The *weight* of i with respect to S is

$$\text{w}_S(i) = \begin{cases} 1, & \text{if } |S| = 1 \\ \sum_{j \in S \setminus \{i\}} \phi_{S \setminus \{i\}}(j, i)\, \text{w}_{S \setminus \{i\}}(j), & \text{otherwise} \end{cases} \tag{4.3}$$

while the *total weight* of S is defined as

$$\text{W}(S) = \sum_{i \in S} \text{w}_S(i)\ . \tag{4.4}$$

Intuitively, $\text{w}_S(i)$ gives us a measure of the overall similarity between vertex i and the vertices of $S \setminus \{i\}$ with respect to the overall similarity among the vertices in $S \setminus \{i\}$.

A subset $S \subseteq V$ is said to be *dominant* if:

1. $\text{w}_S(i) > 0$, for all $i \in S$
2. $\text{w}_{S \cup \{i\}}(i) < 0$, for all $i \notin S$.

The two previous conditions correspond to the two main properties of a cluster [Jain and Dubes, 1988]: the first regards internal homogeneity, whereas the second regards distinctiveness with respect to external objects.

Now, consider the following quadratic program:

$$\begin{aligned} &\text{maximize} && f(\boldsymbol{x}) = \boldsymbol{x}^T A \boldsymbol{x} \\ &\text{subject to} && \boldsymbol{x} \in \Delta_n, \end{aligned} \tag{4.5}$$

where

$$\Delta_n = \{\boldsymbol{x} \in \mathbb{R}^n \,:\, x_i \geq 0 \text{ for all } i \in V \text{ and } \mathbf{1}^T \boldsymbol{x} = 1\} \tag{4.6}$$

is the standard simplex of $\mathbb{R}^n$, and $\mathbf{1}$ is a vector of an appropriate length consisting of unit entries.

In [Pavan and Pelillo, 2007] it is proven that if S is a dominant subset of vertices, then its (weighted) characteristic vector $\mathbf{x}^S$, which is the vector of Δ_n defined as

$$x_i^S = \begin{cases} \frac{w_S(i)}{W(S)}, & \text{if } i \in S \\ 0, & \text{otherwise} \end{cases} \tag{4.7}$$

is a strict local solution of program (4.5). Conversely, if $\boldsymbol{x}$ is a strict local solution of program (4.5) then its support $\sigma(\boldsymbol{x})$ is a dominant set, where the *support* of a vector $\boldsymbol{x} \in \Delta_n$ is defined as the set of indices corresponding to its positive components, that is $\sigma(\boldsymbol{x}) = \{i \in V \,:\, x_i > 0\}$.

By virtue of this result, a dominant set can be found by localizing a local solution of (4.5) and then picking up its support.

Several dominant sets are extracted iteratively by removing the already clustered elements and then maximizing the resulting reduced program (4.5).

4.3.2 Illustrative examples

In order to illustrate the steps followed in the previous subsection to extract the different clusters, some simple examples are analyzed (see Figure 4.3).

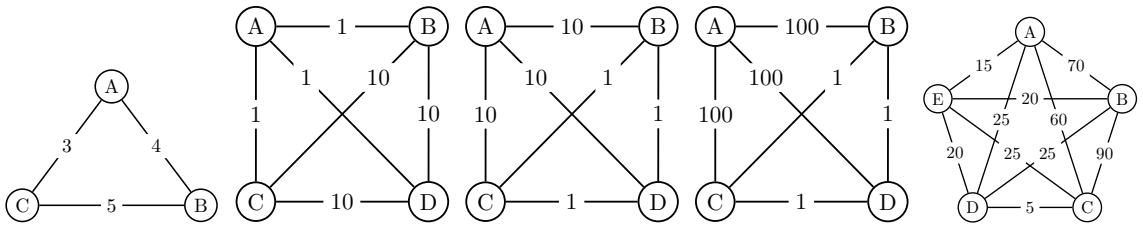


Figure 4.3. Examples of relationships of objects to be grouped, where circles are the objects and lines are the similitude relationship (source: [Pavan and Pelillo, 2007]).

Figure 4.3a illustrates the calculation of (4.3) in a simple manner: $w_{\{A,B,C\}}(C) = \phi_{\{A,B\}}(A,C)\, w_{\{A,B\}}(A) + \phi_{\{A,B\}}(B,C)\, w_{\{A,B\}}(B) = 18$, as:

- $\phi_{\{A,B\}}(A,C) = a_{AC} - \text{awdeg}_{\{A,B\}}(A)$
- $\phi_{\{A,B\}}(B,C) = a_{BC} - \text{awdeg}_{\{A,B\}}(B)$
- $w_{\{A,B\}}(A) = \sum_{j \in \{B\}} \phi_{\{B\}}(j,A)\, w_{\{B\}}(j) = \phi_{\{B\}}(B,A)\, w_{\{B\}}(B)$

- $\mathrm{w}_{\{A,B\}}(B) = \sum_{j\in\{A\}} \phi_{\{A\}}(j,B)\,\mathrm{w}_{\{A\}}(j) = \phi_{\{A\}}(A,B)\,\mathrm{w}_{\{A\}}(A)$
- $\mathrm{awdeg}_{\{A,B\}}(A) = \frac{1}{|\{A,B\}|}\sum_{j=\{A,B\}} a_{Aj} = \frac{1}{2}(a_{AA} + a_{AB})$
- $\mathrm{awdeg}_{\{A,B\}}(B) = \frac{1}{|\{A,B\}|}\sum_{j=\{A,B\}} a_{Bj} = \frac{1}{2}(a_{BA} + a_{BB})$
- $\phi_{\{A\}}(A,B) = \phi_{\{B\}}(B,A) = a_{AB}$
- $\mathrm{w}_{\{A\}}(A) = \mathrm{w}_{\{B\}}(B) = 1$
- $a_{ii} = 0 \forall\, i \in \{A,B,C\}$
- $a_{AB} = a_{BA} = 3$
- $a_{AC} = a_{CA} = 4$
- $a_{BC} = a_{CB} = 5$

Similarly $\mathrm{w}_{\{A,B,C\}}(1) = 10$ and $\mathrm{w}_{\{A,B,C\}}(2) = 16$, which yield $\mathrm{W}(\{A,B,C\}) = 44$.

$\mathrm{w}_S(i)$ is an intuitive measure of the relative similarity of the object i to the rest of the objects of S with respect to the overall similarity of the objects of $S\setminus\{i\}$. For example, the relationship of Figure 4.3b gives $\mathrm{w}^b_{\{A,B,C,D\}}(A) < 0$ while in Figure 4.3c $\mathrm{w}^c_{\{A,B,C,D\}}(A) > 0$ and even stronger in Figure 4.3d $\mathrm{w}^d_{\{A,B,C,D\}}(A) > \mathrm{w}^c_{\{A,B,C,D\}}(A)$. This demeanor can be explained by considering that in Figure 4.3b, object A has a small similitude to the other objects, while the remaining objects have strong similitudes between them, whereas Figure 4.3c shows the opposite, and the similitudes between object A and the rest is even stronger in Figure 4.3d.

Finally Figure 4.3e is added to see how Dominant Set works. The cluster $\{A,B,C\}$ is dominant: the "internal" similitudes (60, 70 and 90) are larger than those between objects belonging to the cluster and others not belonging to it (which have values between 5 and 25).

4.3.3 Computing the Dominant Sets

In order to compute the Dominant Sets a new population dynamics is used, motivated by the analogy with the infection and immunization processes [Rota Bulò and Bomze, 2011]. The selection mechanism governing this dynamics iteratively performs an infection step, which consists of spreading (or suppressing) the most successful (unsuccessful) strategies in the population. The infection phase is then protracted as long as the selected "infective" strategy performs better (or worse, if not extinct) than the average population's payoff. Let $\boldsymbol{e}_i$ be the i-th vector of the canonical base, i.e., a

vector having i-th component equal to 1 and all other components equal to 0. Further, let $\tau_+ = \{i \in V : (A\boldsymbol{x})_i > \boldsymbol{x}^T A\boldsymbol{x}\}$, $\tau_- = \{i \in V : (A\boldsymbol{x})_i < \boldsymbol{x}^T A\boldsymbol{x}\}$ and

$$\mathscr{M}(\boldsymbol{x}) \in \arg\max \left\{(A\boldsymbol{x})_i - \boldsymbol{x}^T A\boldsymbol{x} : i \in \tau_+(\boldsymbol{x})\right\} \cup \left\{-(A\boldsymbol{x})_i + \boldsymbol{x}^T A\boldsymbol{x} : i \in \tau_-(\boldsymbol{x}) \cap \sigma(\boldsymbol{x})\right\}.$$

Then, the dynamics are governed by

$$\boldsymbol{x}^{(t+1)} = \widetilde{\delta}_{\mathscr{S}(\boldsymbol{x}^{(t)})}(\boldsymbol{x}^{(t)})[\mathscr{S}(\boldsymbol{x}^{(t)}) - \boldsymbol{x}^{(t)}] + \boldsymbol{x}^{(t)}, \tag{4.8}$$

where

$$\mathscr{S}(\boldsymbol{x}) = \begin{cases} \boldsymbol{e}_i & i = \mathscr{M}(\boldsymbol{x}) \in \tau_+(\boldsymbol{x}) \\ \frac{x_i}{x_i - 1}(\boldsymbol{e}_i - \boldsymbol{x}) + \boldsymbol{x} & i = \mathscr{M}(\boldsymbol{x}) \in \tau_-(\boldsymbol{x}) \cap \sigma(\boldsymbol{x}) \\ \boldsymbol{x} & \text{otherwise} \end{cases}$$

and

$$\widetilde{\delta}_{\boldsymbol{y}}(\boldsymbol{x}) = \begin{cases} \min\left[1, \frac{(\boldsymbol{x}-\boldsymbol{y})^T A\boldsymbol{x}}{(\boldsymbol{y}-\boldsymbol{x})^T A(\boldsymbol{y}-\boldsymbol{x})}\right] & \text{if } (\boldsymbol{y}-\boldsymbol{x})^T A(\boldsymbol{y}-\boldsymbol{x}) < 0 \\ 1 & \text{otherwise.} \end{cases}$$

This evolution process exhibits a number of nice properties [Rota Bulò and Bomze, 2011], but in particular it is computationally very efficient, as each iteration has linear time complexity as opposed to the quadratic complexity of the replicator dynamics used in [Pavan and Pelillo, 2007].

4.4 Detection framework

The detection framework has to decide wether a behavior is abnormal or not to alert the operator of the surveillance system. As was pointed out by [Nayak et al., 2011], the easiest way to know if a behavior is abnormal is to train the system to detect normal behavior and consider as abnormal any behavior that do not fit the normal clusters. The most recurrent behaviors obtained from the training set are the normal behaviors, as usually, abnormal situations occurs infrequently.

4.4.1 Component selection

The input data for the behavior analysis are commonly the output of the tracking step. Several approaches in the literature use the whole trajectory of a person as input, deciding if it is normal or abnormal [Hu et al., 2006]. However, Jiang et al. [Jiang et al., 2011] point out that an anomaly may not correspond to a whole trajectory, but only to a part of it. That is why they use atomic events to detect smaller anomalies.

The data used to train and run the clusterization step are composed of the location

of the object of interest, its direction and its speed. These data are easily available as the output of any tracking video system and have already been used in other behavior analysis models [Hu et al., 2006, Jiang et al., 2011]. The behavior are classified as normal or not using exclusively these four values, thus, the framework will trigger an alarm when a detected behavior is not similar in location, speed **and** direction to any known one. In real life, abnormal situations would be situations where the tracked objects takes unusual paths (i.e. people entering a room through the window), walking faster than usual or stopping unexpectedly (i.e. a car accident in a freeway) or going in the opposite direction as most people (i.e. entering in a theater through the exit door).

Following Jan [Jan, 2004], the top point of the head is used as input value, instead of using the centroid of the blobs detected in the motion detection step. This allows the model to solve problems of false speed calculation due to occlusions or people not entirely in the camera vision area.

4.4.1.1 Speed and direction estimation

After any simple tracking model, the results obtained correspond to simple location of the studied object in the image for each frame. From these data, an estimation of the speed and direction of the object must be evaluated. To make it robust enough while keeping it as fast as possible, in each frame the position of the object tracked is considered to follow a linear regression. That way, it is easier to estimate the speed and direction.

The slope of the movement in the image at frame T can be obtained following:

$$slope_T = \frac{N \cdot \sum_{t=T-n}^{T} x_t y_t - \sum_{t=T-n}^{T} x_t \sum_{t=T-n}^{T} y_t}{N \cdot \sum_{t=T-n}^{T} x_t^2 - (\sum_{t=T-n}^{T} x_t)^2}, \tag{4.9}$$

where, (x_t, y_t) are the x and y coordinates of the tracked object in image at frame t, and n is the number of coordinates used to calculate the slope. Simplifying, using Equation (4.10), the slope is calculated as Equation (4.11) and can be easily updated using Equation (4.12).

$$\begin{aligned} S_{x,T} &= \sum_{t=T-n}^{T} x_t, \\ S_{y,T} &= \sum_{t=T-n}^{T} y_t, \\ S_{xx,T} &= \sum_{t=T-n}^{T} x_t^2, \\ S_{xy,T} &= \sum_{t=T-n}^{T} x_t y_t. \end{aligned} \tag{4.10}$$

$$slope_T = \frac{N \cdot S_{xy,T} - S_{x,T} \cdot S_{y,T}}{N \cdot S_{xx,T} - (S_{x,T})^2} \tag{4.11}$$

$$\begin{aligned} S_{x,T} &= S_{x,T-1} + x_T - x_{T-n}, \\ S_{y,T} &= S_{y,T-1} + y_T - y_{T-n}, \\ S_{xx,T} &= S_{xx,T-1} + x_T^2 - x_{T-n}^2, \\ S_{xy,T} &= S_{xy,T-1} + x_T * y_T - x_{T-n} * y_{T-n}. \end{aligned} \tag{4.12}$$

Once obtained, the slope, the speed and direction can be calculated as follows:

$$\begin{aligned} speed_T &= (x_T - x_{T-n}) \cdot \sqrt{1 + slope_T^2}, \\ direction_T &= \arctan(slope_T). \end{aligned} \tag{4.13}$$

Note that speed value contains information about the heading, i.e. negative value model movement for one heading and positive in the opposite.

4.4.2 Model training

4.4.2.1 Normalization

To avoid the risk of awarding an excessive importance to any of the four input dimensions, a normalization is made using the training data set. The mean μ_k and variation σ_k^2 of each dimension k are analyzed:

$$\begin{aligned} \mu_k &= \tfrac{1}{D_{tr}} \sum_{i=1}^{D_{tr}} d_{i,k}, \\ \sigma_k^2 &= \mu_{k^2} - \mu_k^2. \end{aligned} \qquad \forall k = \{x, y, speed, direction\} \tag{4.14}$$

where D_{tr} is the size of the training data set, $d_{i,k}$ is the value of dimension k of the i^{th} input vector and μ_{k^2} is the mean of the squared values of dimension k.

All the input data, from both training and test data sets are then normalized using the values in (4.14):

$$d_{i,k}^* = (d_{i,k} - \mu_k)/\sigma_k \quad \forall k, i = \{1, \cdots, D_{tr} + D_{te}\} \tag{4.15}$$

where $d_{i,k}^*$ is the normalized value of dimension k of i^{th} vector and D_{te} the size of input test data set.

4.4.2.2 Similitude calculation

Once all the values are normalized, the similarity a_{ij} between each two data points i and j is calculated. As has been seen in [Pavan and Pelillo, 2007], it can be obtained through:

$$\begin{aligned} a_{ij} &= \exp(-||\vec{F}(i) - \vec{F}(j)||_2^2 / c^2) \\ &= \exp(-\sum_k \text{abs}((d^*_{k,i})^2 - (d^*_{k,j})^2) / c^2) \end{aligned} \tag{4.16}$$

where c is a positive real number which affects the decreasing rate of a, and $\vec{F}$ is the normalized 4D vector of each element of test and training sets.

4.4.2.3 Parameter choice

The only parameter of the proposed approach is the similitude decay rate c. The ideal value of c is task specific and depends heavily on the data received. Figure 4.4 plots the total error rate (sum of false positive rate and false negative rate) of the approach in two test sets (IIT1 and ICAI1 as described in the experimental section) as parameter c varies between 1 and 200. The optimal values are quite different for the two test sets and are very sensitive to variation in the IIT1 dataset. Note that as c increases, the clusters become larger (as seen by the weight $W_S = \boldsymbol{x}^T A \boldsymbol{x}$ of the first cluster) and thus the false-negative rate decreases. This, however, significantly increases the false positive rate.

While there is no fixed rule for selecting c, experimentally it has been seen that good values are obtained when the first dominant set contains around 15% of the data. However, since in security applications false negatives are considered more dangerous than a false negative, the model is set to over-segment the data even more, selecting a slightly smaller than optimal parameter c, which produces a first dominant set covering 10% of the data. This decreases the false negative rate, while increasing the false positive rate by a small amount.

4.4.3 Abnormal behavior detection

Once the similitudes between the elements of the training data set have been obtained, the Dominant Sets clustering method can be used to obtain the different behaviors captured by the camera. In an unsupervised scenario, what potentially dangerous behavior is cannot be known. Instead, normal behavior is considered to happen frequently, so common events are considered to be normal people behaviors. These events are then modeled with Dominant Sets, while any not learned event is detected as an abnormal behavior.

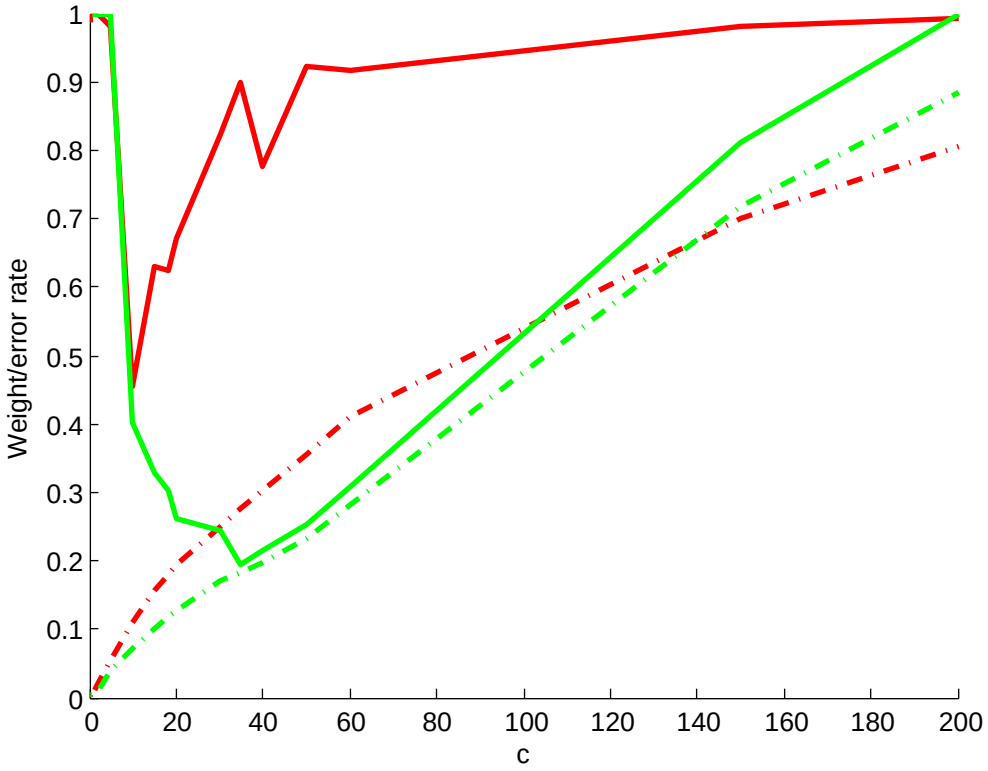


Figure 4.4. Weight of the first cluster (dashdotted line) and error (solid line) rates as a function of c in cases 5 (red) and 7 (green).

The Dominant Sets framework returns a set of clusters containing all the training data. Each of these clusters has a different size, which makes it simple to differentiate common behaviors from less frequent events. Clusters containing less than 1% of the training data are deleted, while the rest are used as the model for normal behaviors.

Each new datum i obtained by the system is normalized as seen in Equation (4.15) and then analyzed to see if it belongs to any of the clusters modeling normal behaviors following $w_{S\cup\{i\}}(i) > 0$. In [Pavan and Pelillo, 2004], they demonstrate that it can be calculated in a easier way:

$$w_{S\cup\{i\}}(i) > 0 \Leftrightarrow \sum_{h\in S} (a_{hi}\boldsymbol{x}_h) > f(\boldsymbol{x}) = \boldsymbol{x}^T A\boldsymbol{x} \tag{4.17}$$

where S is the subspace containing the data of the cluster, a_{hi} is the similarity between new data i and data h, $\boldsymbol{x}$ is the support vector obtained from (4.5) and $\boldsymbol{x}(h)$ is the h^{th} value of $\boldsymbol{x}$.

If condition (4.17) is not met in any of the clusters obtained with the training set, the person analyzed is acting strangely and the behavior is tagged as abnormal.

4.4.4 Illustrative example

This subsection presents a simple example of behavior analysis, in order to understand the functioning of the model in an image of $5 * 5px$. Figure 4.5 shows the trajectories of the training set.

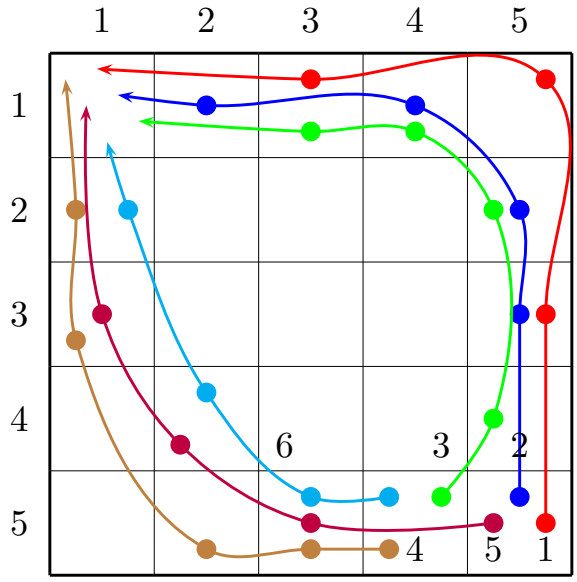


Figure 4.5. Representation of the trajectories of the training set. Each square represents a pixel, and not all the positions are situated in the middle of the pixel to make the trajectories clearer.

The speed calculus can be simplified to a simpler $\Delta_x(t) = x(t) - x(t-1)$ and $\Delta_y(t) = y(t) - y(t-1)$ with $\Delta_x(0) = 0$ and $\Delta_y(0) = 0$, obtaining the data of Table 4.1. The normalization of the data gives:

- $\mu_k \approx \{2.970, 2.848, -0.636, -0.727\}$ for $k = \{x, y, \Delta_x, \Delta_y\}$ respectively.
- $\sigma_k^2 \approx \{2.575, 2.856, 0.595, 0.623\}$ for $k = \{x, y, \Delta_x, \Delta_y\}$ respectively.

The similarity matrix is then constructed using (4.16), leaving $c = 1$ to simplify, and a set of 10 Dominant Sets is obtained following Section 4.3. As an example, the similitude between position $tr1b$ and test position $ter3d$ is calculated as: $a_{tr1b,tr3c} \approx \exp(-(|1.265^2 - 0.642^2| + |0.090^2 - (-1.094)^2| + (|0.825^2 - (-0.471)^2| + |-1.613^2 - (-0.346)^2|)) \approx 0.0049$. Other similitudes are easily calculated, as $a_{tr1a,tr2a} = 1$. The clusters created are shown in Figure 4.6. Although the representation is a $2D$ figure, it has to be noted that the positions are $4D$ elements.

Two trajectories, shown in Figure 4.7, are analyzed as test data to see the performance of the abnormal behavior detection model. Table 4.2 presents their data and normalization values.

The similitude between each $4D$-position of the test set and each one of the training set is analyzed. Those values are then used to decide if the behavior analyzed is normal

Table 4.1. Training data obtained from tracking and its normalization.

#	x	y	Δ_x	Δ_y	x^*	y^*	Δ_x^*	Δ_y^*
tr1a	5	5	0	0	1.265	1.273	0.825	0.922
tr1b	5	3	0	-2	1.265	0.090	0.825	-1.613
tr1c	5	1	0	-2	1.265	-1.094	0.825	-1.613
tr1d	3	1	-2	0	0.019	-1.094	-1.768	0.922
tr1e	1	1	-2	0	-1.228	-1.094	-1.768	0.922
tr2a	5	5	0	0	1.265	1.273	0.825	0.922
tr2b	5	3	0	-2	1.265	0.090	0.825	-1.613
tr2c	5	2	0	-1	1.265	-0.502	0.825	-0.346
tr2d	4	1	-1	-1	0.642	-1.094	-0.471	-0.346
tr2e	2	1	-2	0	-0.604	-1.094	-1.768	0.922
tr2f	1	1	-1	0	-1.228	-1.094	-0.471	0.922
tr3a	4	5	0	0	0.642	1.273	0.825	0.922
tr3b	5	4	1	-1	1.265	0.681	2.121	-0.346
tr3c	5	2	0	-2	1.265	-0.502	0.825	-1.613
tr3d	4	1	-1	-1	0.642	-1.094	-0.471	-0.346
tr3e	3	1	-1	0	0.019	-1.094	-1.768	0.922
tr3f	1	1	-2	0	-1.228	-1.094	-0.471	0.922
tr4a	4	5	0	0	0.642	1.273	0.825	0.922
tr4b	3	5	-1	0	0.019	1.273	-0.471	0.922
tr4c	2	5	-1	0	-0.604	1.273	-0.471	0.922
tr4d	1	3	-1	-2	-1.228	0.090	-0.471	-1.613
tr4e	1	2	0	-1	-1.228	-0.502	0.825	-0.346
tr4f	1	1	0	-1	-1.228	-1.094	0.825	-0.346
tr5a	5	5	0	0	1.265	1.273	0.825	0.922
tr5b	3	5	-2	0	0.019	1.273	-1.768	0.922
tr5c	2	4	-1	-1	-0.604	0.681	-0.471	-0.346
tr5d	1	3	-1	-1	-1.228	0.090	-0.471	-0.346
tr5e	1	1	0	-2	-1.228	-1.094	0.823	-1.613
tr6a	4	5	0	0	0.642	1.273	0.825	0.922
tr6b	3	5	-1	0	0.019	1.273	-0.471	0.922
tr6c	2	4	-1	-1	-0.604	0.681	-0.471	-0.346
tr6d	1	2	-1	-2	-1.228	-0.502	-0.471	-1.613
tr6e	1	1	0	-1	-1.228	-1.094	0.825	-0.346

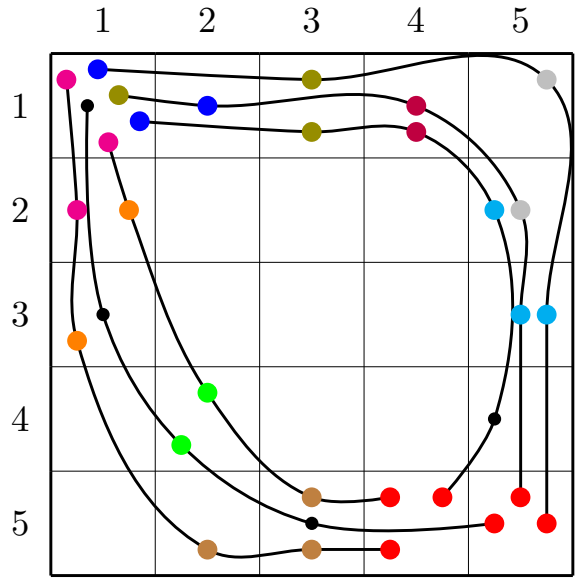


Figure 4.6. Representation of the Dominant Sets obtained in different colors (in order: red, blue, green, brown, purple, cyan, magenta, orange, light gray and olive). Black dots correspond to unclustered positions

or not, using (4.17). Note that $\boldsymbol{x}^T A\boldsymbol{x}$ may be calculated offline for each cluster. To illustrate how to proceed to classify a new position, the similitudes of test position $te2f$ to the fourth (brown) and sixth (cyan) cluster are calculated. The characteristic vectors $\boldsymbol{x}^{DS4}$ and $\boldsymbol{x}^{DS6}$ of these Dominant Sets, obtained from Equation (4.5), are:

- $$\boldsymbol{x}_i^{DS4} \approx \begin{cases} 0.396086, & \text{if } i \in \{tr4b\} \\ 0.208052, & \text{if } i \in \{tr4c\} \\ 0.395862, & \text{if } i \in \{tr6b\} \\ 0, & \text{otherwise} \end{cases}$$

- $$\boldsymbol{x}_i^{DS6} \approx \begin{cases} 0.387621, & \text{if } i \in \{tr1b\} \\ 0.387361, & \text{if } i \in \{tr2b\} \\ 0.225018, & \text{if } i \in \{tr3c\} \\ 0, & \text{otherwise} \end{cases}$$

That gives $(\boldsymbol{x}^{DS4})^T A\boldsymbol{x}^{DS4} = 0.537066$ and $(\boldsymbol{x}^{DS6})^T A\boldsymbol{x}^{DS6} = 0.546033$. Finally, the values of the left term of the inequality with element $te2f$ are:

- $\sum_{h \in DS4} a_{h,te2f}\boldsymbol{x}_h^{DS4} \approx 1 * 0.396086 + 0.678159 * 0.208052 + 1 * 0.395862 \approx 0.93304 >$ 0.537066, thus it belongs to the fourth Dominant Set cluster.

- $\sum_{h \in DS6} a_{h,te2f}\boldsymbol{x}_h^{DS6} \approx 1.57374 \cdot 10^{-005} * 0.387621 + 1.57374 \cdot 10^{-005} * 0.387361 + 2.73255 \cdot 10^{-006} * 0.225018 \approx 1.281 \cdot 10^{-005} < 0.546033$, thus it does not belong to the sixth Dominant Set cluster.

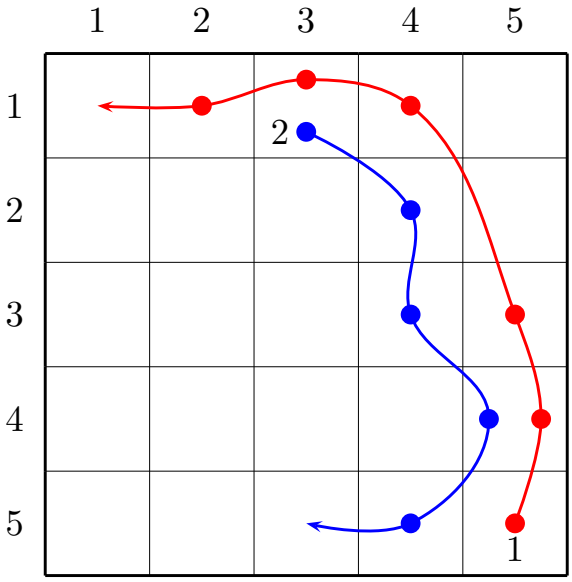


Figure 4.7. Representation of the trajectories of the test set.

The complete results are depicted in Figure 4.8. It can be seen that nearly all the "abnormal" (second) trajectory is correctly tagged as abnormal as its positions do not belong to any cluster, while the "normal" (first) trajectory is completely tagged as normal as all its positions are included in one DS.

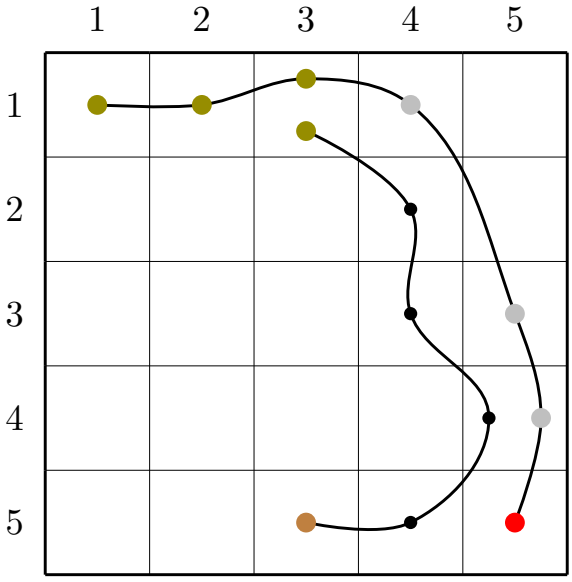


Figure 4.8. Results of the illustrative example, with each position in the color of the corresponding DS cluster.

4.5 Experimental results

The proposed abnormal behavior detection approach was tested on five video sequences. These video sequence are divided in training and test sets, where the training

Table 4.2. Test data obtained from tracking and its normalization.

#	x	y	Δ_x	Δ_y	x^*	y^*	Δ_x^*	Δ_y^*
te1a	5	5	0	0	1.265	1.273	0.825	0.922
te1b	5	4	0	-1	1.265	0.681	0.825	-0.346
te1c	5	3	0	-1	1.265	0.090	0.825	-0.346
te1d	4	1	-1	-2	0.642	-1.094	-0.471	-1.613
te1e	3	1	-1	0	0.019	-1.094	-0.471	0.922
te1f	2	1	-1	0	-0.604	-1.094	-0.471	0.922
te1g	1	1	-1	0	-1.228	-1.094	-0.471	0.922
te2a	3	1	0	0	0.019	-1.094	0.825	0.922
te2b	4	2	1	1	0.642	-0.502	2.121	2.189
te2c	4	3	0	1	0.642	0.090	0.825	2.189
te2d	5	4	1	1	1.265	0.681	2.121	2.189
te2e	4	5	-1	1	0.642	1.273	-0.471	2.189
te2f	3	5	-1	0	0.019	1.273	-0.471	0.922

sets contains at least one of the abnormal behavior pattern a surveillance system has to detect.

IndoorGTTest2 : short indoor video with people walking from left to right and from right to left, available from IBM PeopleVision webpage[1] [Hampapur et al., 2003].

AVSS PV Easy : video from the 2007 IEEE International Conference on Advanced Video and Signal based Surveillance[2]. The video shows a road junction where a very busy road and a rarely used one converge. A van parks in a non-parking section. The training set corresponds to vehicle movement before the van appears. The test set corresponds to the vehicles after the van appears, including the van itself.

IIT1 : indoor video showing several people following the same trajectory many times for several minutes, then walking more freely: going backwards, running, unexpectedly stopping or walking in restricted areas. The training set corresponds to the first minutes, while the test set, divided into normal and abnormal behaviors, corresponds to the rest of the sample video.

ICAI1 : outdoor video of a courtyard from a bird's eye view. There are clear normal paths between the different four access doors, and some people taking other abnormal trajectories. The abnormal behaviors are all included in a test set of abnormal behaviors. The rest of the data is divided into training (till frame 32500) and test sets (from frame 32501)

UCSD Anomaly Detection Dataset (UCSD) : outdoor videos of a pedestrian avenue. People are walking from top to bottom or from bottom to top. The abnormal

[1] www.research.ibm.com/peoplevision
[2] http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html

behaviors are people on bikes or skaters moving faster than the people on foot[3] [Mahadevan et al., 2010, Li et al., 2013].

The data sets used here are obtained manually from these five videos and can be accessed from the author's webpage[4]. The values used are the x and y coordinates of the top-left corner of the tracked object of interest, the speed of the object and the direction of the trajectory. Speed and direction are calculated as explained in Equation (4.13). All four values are normalized following (4.15).

UCSD is usually taken as a comparison dataset for crowd abnormal behavior detectors, such as [Mahadevan et al., 2010, Li et al., 2013, Cong et al., 2011, Cong et al., 2013, Adam et al., 2008, Mehran et al., 2009, Kim and Grauman, 2009]. The aim of the Dominant Sets abnormalities detector is set to detect abnormalities of a single object in the scene, not to analyze the crowd behavior. Nevertheless, Appendix C provides a comparison of the model herein presented and the abnormal behavior detection in crowded scenes models to show the robustness of the model in different applications.

Figure 4.9 presents the different training sets used in these experiments.

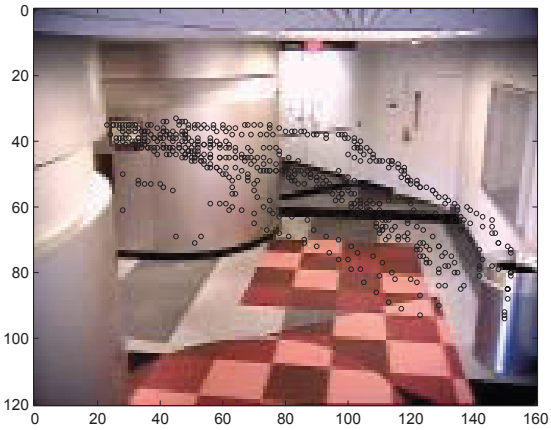
4.5.1 KNN, mixture of Gaussians and Fuzzy K-Means

A comparison was carried out of the proposed Dominant Sets-based framework and three classical approaches based on a mixture of Gaussians, a KNN density estimation and an adaptation of the two-step Fuzzy K-Means presented in [Hu et al., 2006]. As the values are normalized before running the Dominant Sets method presented, the same normalized values are used for these two algorithms.

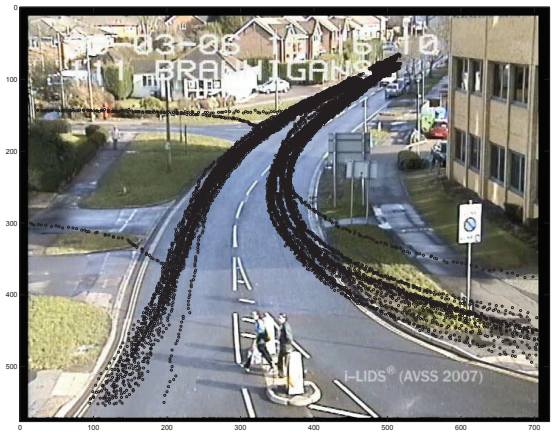
In the first case, the training set was clustered using K-means to choose an optimum number of Gaussians to model the normal space. For each set, fifteen random initializations for each value of K were performed. That is, fifteen different K-means results for each value of K, with K varying from 1 to 30, were computed. The total sum of distances between each point of the training set and the nearest center is plotted as a function of the number of clusters K as in Figure 4.10. With this graphic, an optimal number of clusters can be chosen for the mixture of Gaussians that will model the normal behavior space represented in the input values. The optimal value is chosen from the elbow of the curve.

Once the training set has been modeled with a mixture of Gaussians, each point of the test set is compared to each Gaussian of the mixture. The behavior of the person in that instant is considered to be normal if it is considered to belong to any cluster, and abnormal otherwise. Assuming normal distribution in the clusters, the $\chi^2_4(0.05)$ value is used as a threshold to evaluate if the new data belong to any cluster or not following

[3] http://www.svcl.ucsd.edu/projects/anomaly/
[4] http://iit.upcomillas.es/malvar/DominantSets.php

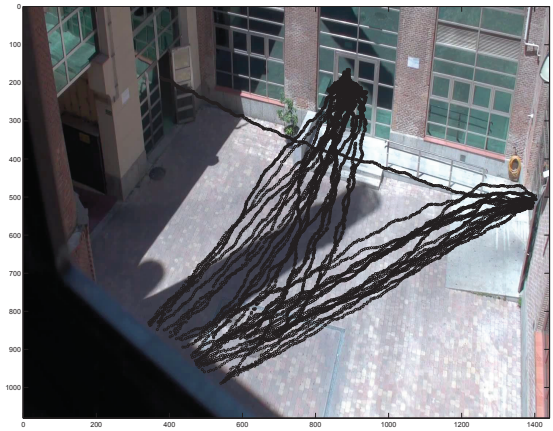


(a) IndoorGTTest2

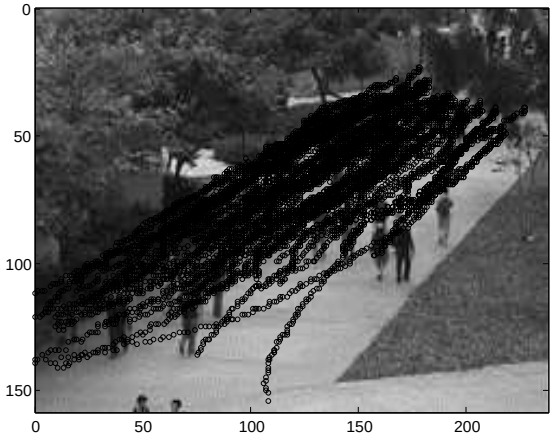


(b) AVSS PV Easy

(c) IIT1



(d) ICAI1

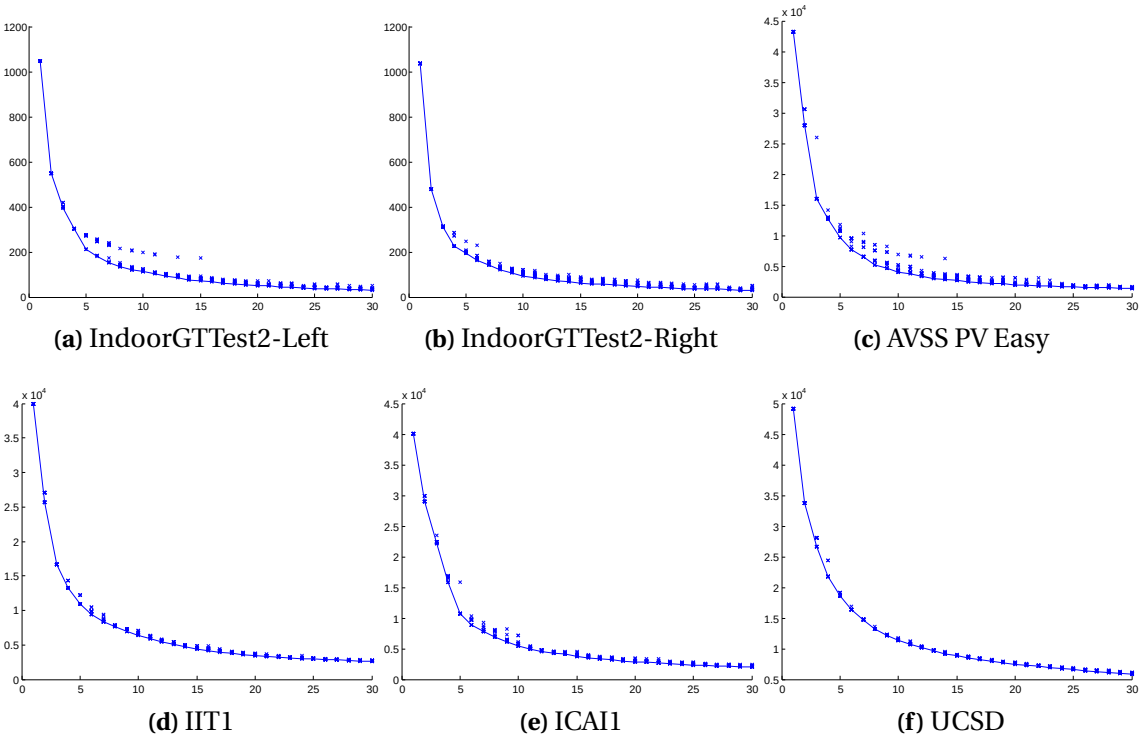


(e) UCSD

Figure 4.9. Training sets of different videos.

Equation (4.18), where $\boldsymbol{x}^*$ is the normalized value analyzed, $\boldsymbol{\mu_G}$ is the mean and $\boldsymbol{\Sigma_G}$ is the covariance matrix of Gaussian G.

$$(\boldsymbol{x}^* - \boldsymbol{\mu_G})' {\boldsymbol{\Sigma_G}}^{-1} (\boldsymbol{x}^* - \boldsymbol{\mu_G}) \leq \chi^2_4(0.05) \tag{4.18}$$



(a) IndoorGTTest2-Left **(b)** IndoorGTTest2-Right **(c)** AVSS PV Easy

(d) IIT1 **(e)** ICAI1 **(f)** UCSD

Figure 4.10. Total distance of the points to the cluster as a function of K for training sets of videos IndoorGTTest2, AVSS PV Easy, IIT1, ICAI1 and UCSD.

A second comparison is carried out using the K nearest neighbors algorithm. The distance to the fifth nearest neighbor of the training set is calculated for each point in the test set. The bigger this distance is, the less normal the behavior is considered to be. In order to set a correct threshold for this method to determine if it is normal or not, the distance of each point of the training set to its neighbors is analyzed. Once the mean distance and standard deviation to the fifth nearest neighbor have been obtained, the threshold is set to the mean plus three times the standard deviation. Figure 4.11 shows an example of the values of the distances of each point of training and test sets, and the threshold obtained.

The two-step Fuzzy K-Means (FKM) clusterization presented in [Hu et al., 2006] proposes firstly performing a spatial-based clustering using Fuzzy K-Means. In each of these clusters, a second clustering is carried out based on temporal data. As said before, K-Means needs to know a priori the number of clusters the data are composed of. As proposed by the authors, the Tightness and Separation Criterion (TSC) [Xie and Beni, 1991] is used in both steps to decide the number of clusters.

4.5.2 Results

The results of the Dominant Sets method presented, as well as the results using KNN algorithm and Gaussians Mixture algorithm are shown in Figures 4.12 to 4.19. As

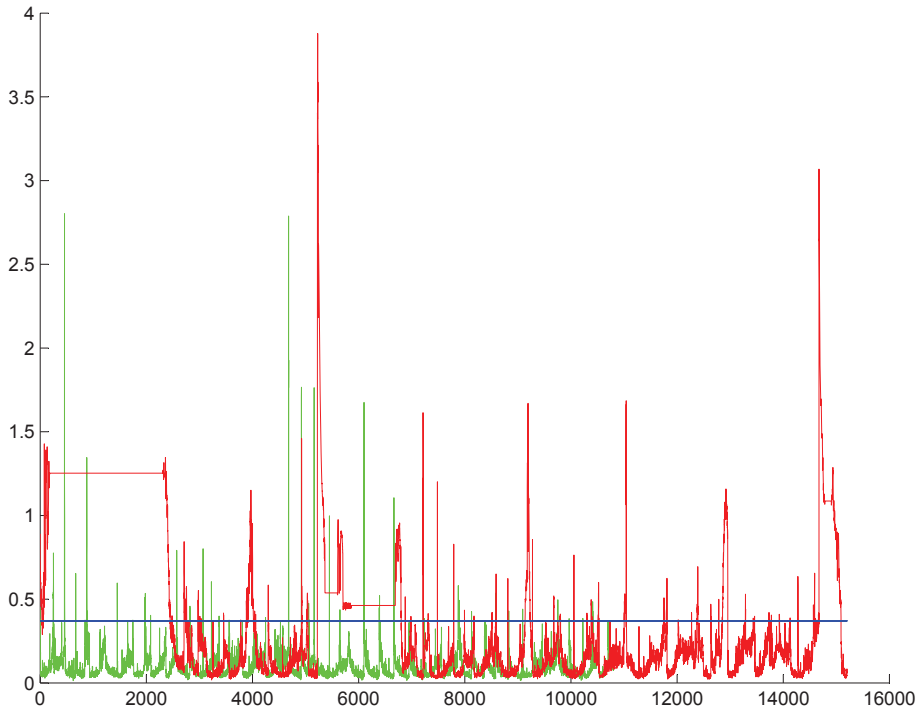
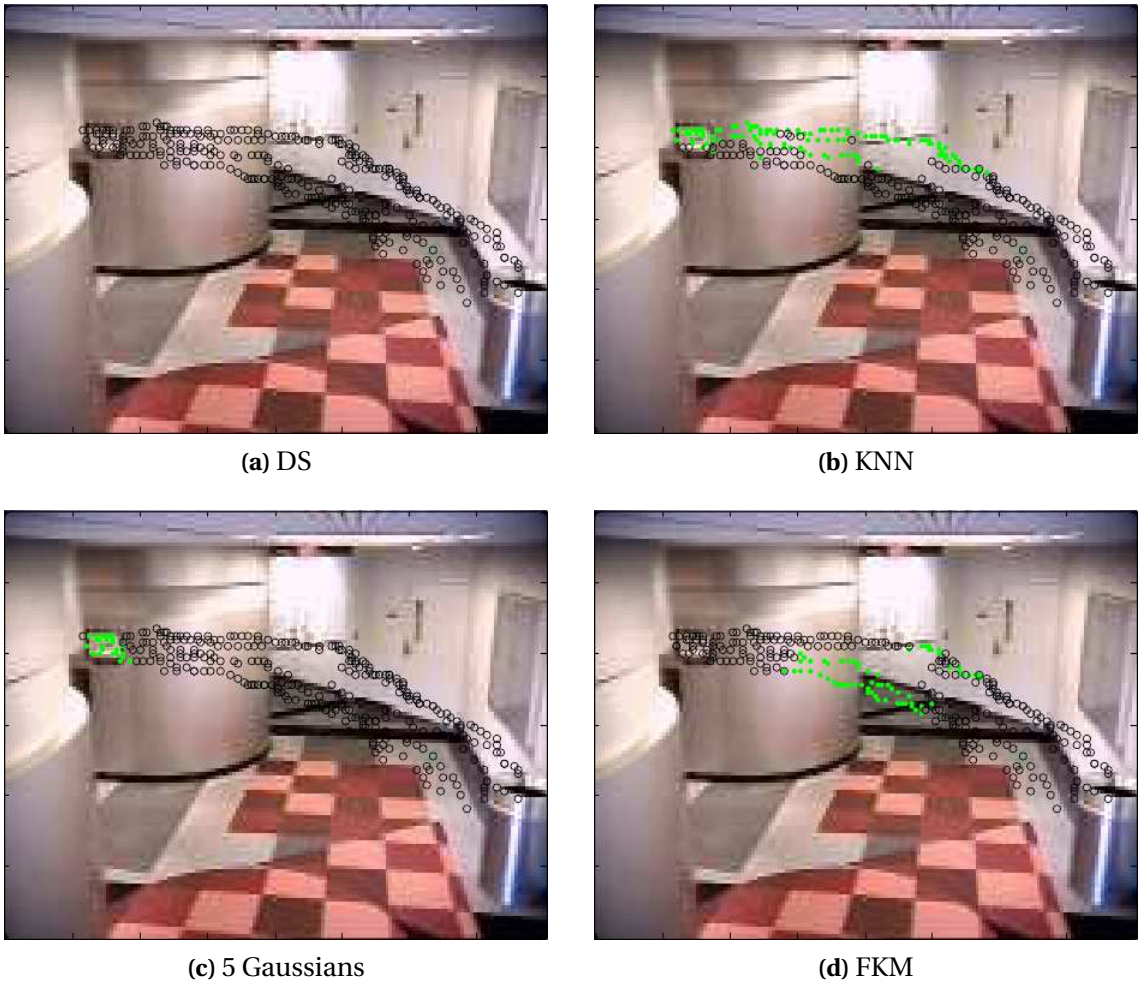


Figure 4.11. Distance to the 5^{th} NN for each point of training (green) and test (red) set and threshold (blue) in the video AVSS PV Easy.

previously explained, the number of Gaussians is chosen taking the optimal number of clusters to model the space described in the training set in the first case, and a total number of Gaussians equal to the number of clusters obtained with Dominant Sets (DS). In the case of the IndoorGTTest video, the data set was too short and the mixture of Gaussians test was only performed with an optimal number of Gaussians. Green dots correspond to normal behavior previously learned, while black circles correspond to abnormal behavior. Note that, although the data are shown as points in a 2D image, each sample corresponds to a 4D point, but speed and direction coordinates are not displayed.

These images show a general outline of the different methods' performance. Table 4.3 presents further details of each case study, with the size of training and test sets, as well as the number of samples that have been manually considered abnormal. Note that the first two cases are added to show the performance in simple abnormal behavior detection, as all the people in the training set walk in one direction while in the test set they walk in the opposite one. They are thus not meant to model realistic behaviors, but to analyze performance in easy cases.

The results of each method are shown in Tables 4.4 and 4.5. Since the results of MoG and FKM methods depend on the initialization, the number of false-negative (FN), people having an abnormal behavior labeled as normal, and false-positive (FP), normal behaviors labeled as abnormal, errors of this method correspond to the mean error obtained through twenty executions. The *Op. MoG* column corresponds to the results with an optimum number of Gaussians using the K-means results as shown in



(a) DS **(b)** KNN

(c) 5 Gaussians **(d)** FKM

Figure 4.12. People moving right set detection results with all methods in the IndoorGTTest2 video with people moving left as the training set (case 1). Green dots are normal behaviors while black circles are abnormal detections.

Figure 4.10. Results creating as many Gaussians as clusters are constructed in DS and shown in the *DS MoG* column. Figure 4.20 shows the results in the ROC space in case 3. The overall error ratio of DS is 1.5 points lower than FKM which is the best of the four comparison algorithms.

It can be seen that the different methods present different performances depending on the input data. It has to be taken into account that the number of Gaussians for the mixture of Gaussians methods is not automatically chosen. In the first case, a Pareto curve (see Figure 4.10) has to be obtained previously and then the number of Gaussians can be determined. In these cases, a value around 6 seemed a correct approximation, but that may vary in other more complex environments. Of course, as the choice of the number of Gaussians in the last column is obtained taking the number of clusters in DS, this number cannot be automatically selected without previously executing DS. Nonetheless, these results are presented for comparative purposes.

Several conclusions can be drawn from Tables 4.4, 4.5 and Figure 4.20. First, it can be

Table 4.3. Study of cases.

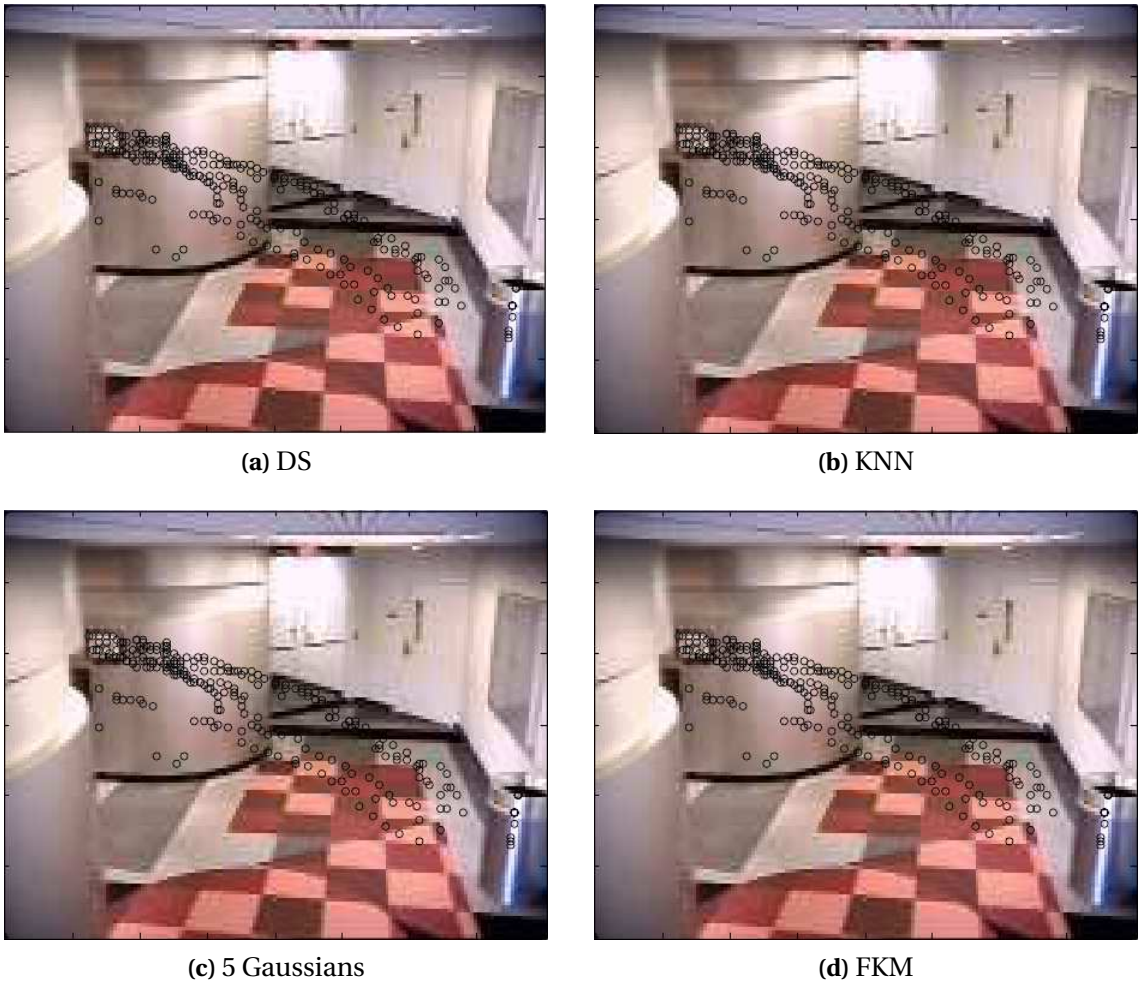
case	video	training set size	test set size	abnormal samples
1	IndoorGTTest2	262	301	301
2	IndoorGTTest2	301	262	262
3	AVSS	10821	15203	4445
4	IIT1	9991	1602	0
5	IIT1	9991	886	587
6	ICAI1	10032	3191	0
7	ICAI1	10032	3125	2866
8	UCSD	12803	2803	454

Table 4.4. # of false positives and negatives for each case and each method.

case	DS		KNN		Op. MoG		DS MoG		FKM	
	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP
1	0	-	114	-	16	-	NA	NA	0	-
2	0	-	0	-	0	-	NA	NA	0	-
3	65	728	23	649	2333	904	1119	1802	593	334
4	-	90	-	20	-	31	-	28	-	164
5	130	21	204	11	215	21	96	26	404	30
6	-	211	-	1006	-	592	-	1069	-	352
7	175	41	20	47	63	39	37	61	60	46
8	84	537	142	344	94	339	72	487	239	294

Table 4.5. % of wrong classification of abnormal (A) and normal (N) behaviors for each case and each method.

case	DS		KNN		Op. MoG		DS MoG		FKM	
	A	N	A	N	A	N	A	N		
1	0	-	37.9	-	5.3	-	NA	NA	0	-
2	0	-	0	-	0	-	NA	NA	0	-
3	1.5	6.8	0.5	6.0	52.5	8.4	25.2	16.8	13.3	3.1
4	-	5.6	-	1.2	-	1.9	-	1.7	-	10.2
5	22.1	7.0	34.8	3.7	36.6	7.0	16.4	8.7	68.9	10.0
6	-	6.6	-	31.4	-	18.6	-	33.5	-	11.0
7	6.1	15.8	0.7	18.1	2.2	15.1	1.3	23.6	2.1	2.0
8	18.5	22.9	31.3	14.6	20.7	14.4	15.9	20.7	52.6	12.5



(a) DS **(b)** KNN

(c) 5 Gaussians **(d)** FKM

Figure 4.13. People moving left set detection results with all methods in the IndoorGTTest2 video with people moving right as the training set (case 2). Green dots are normal behaviors while black circles are abnormal detections.

seen that the Dominant Sets method presented and FKM method are the only methods that do a perfect job in the first two cases. These cases correspond to simple abnormal behavior detection, as all the people in the training set walk in one direction while in the test set they walk in the opposite one. Secondly, although in case 4 DS has a higher error rate than the other methods (except FKM), the FP error remains constant in cases 1 to 6, and is significantly smaller in case 6. The third case shows poor results for Mixture of Gaussians methods compared with DS, FKM and KNN , the latter having a better performance. In case 5, the results obtained in DS are similar to those of the other methods. Cases 7 and 8 show a higher FN error in the method presented, while FP, although high, is still lower than the other comparison methods. Figure 4.21 shows detailed information of FN and FP of DS results. Finally, it has to be noted that the errors made by DS tend to remain lower than 10%, and in the cases with a higher ratio the error is similar to or lower than the error made by the other methods.

The extra FN detections in case 3 correspond to two events. On the one hand, it

(a) DS

(b) KNN

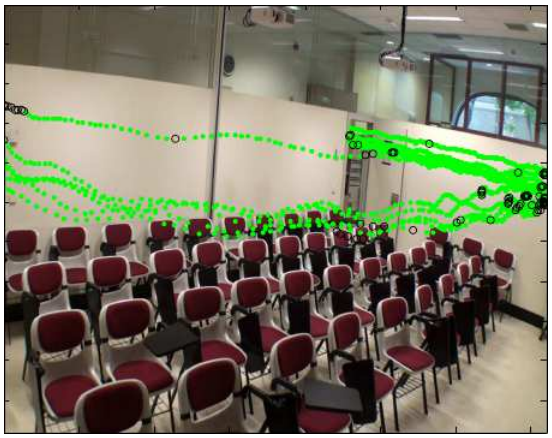
(c) 6 Gaussians

(d) 21 Gaussians

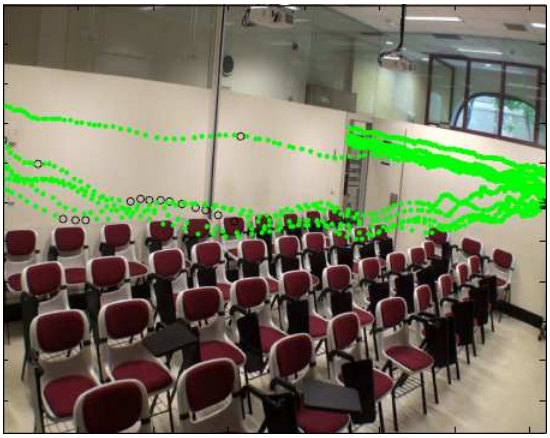
(e) FKM

Figure 4.14. Normal behavior test set detection results with all methods in AVSS PV Easy (case 3). Green dots are normal behaviors while black circles are abnormal detections.

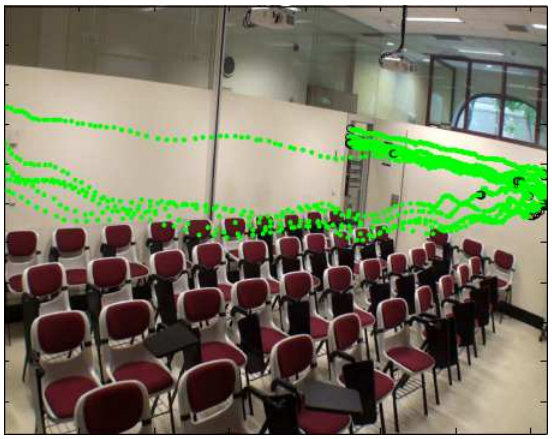
detects the behavior of the van before stopping as acceptable, while it has been manually labeled as abnormal due to the proximity to the sidewalk. On the other hand, a few elements corresponding to the first/last moments of vehicles turning into/out of the lateral street are labeled by DS as normal while the ground truth considers them as



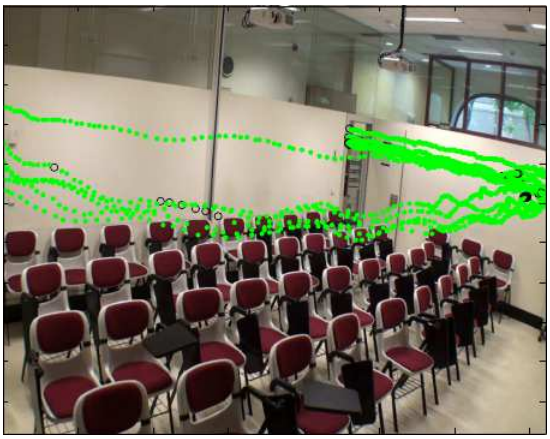
(a) DS



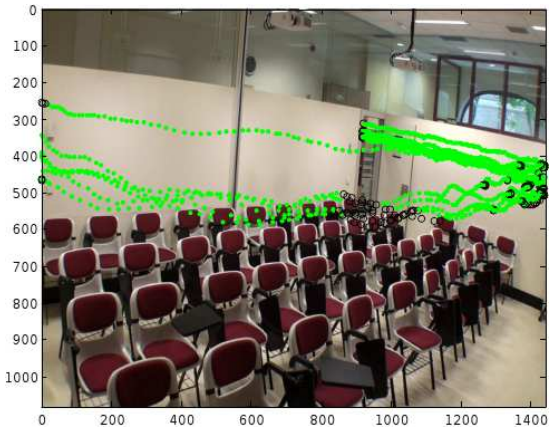
(b) KNN



(c) 6 Gaussians



(d) 26 Gaussians

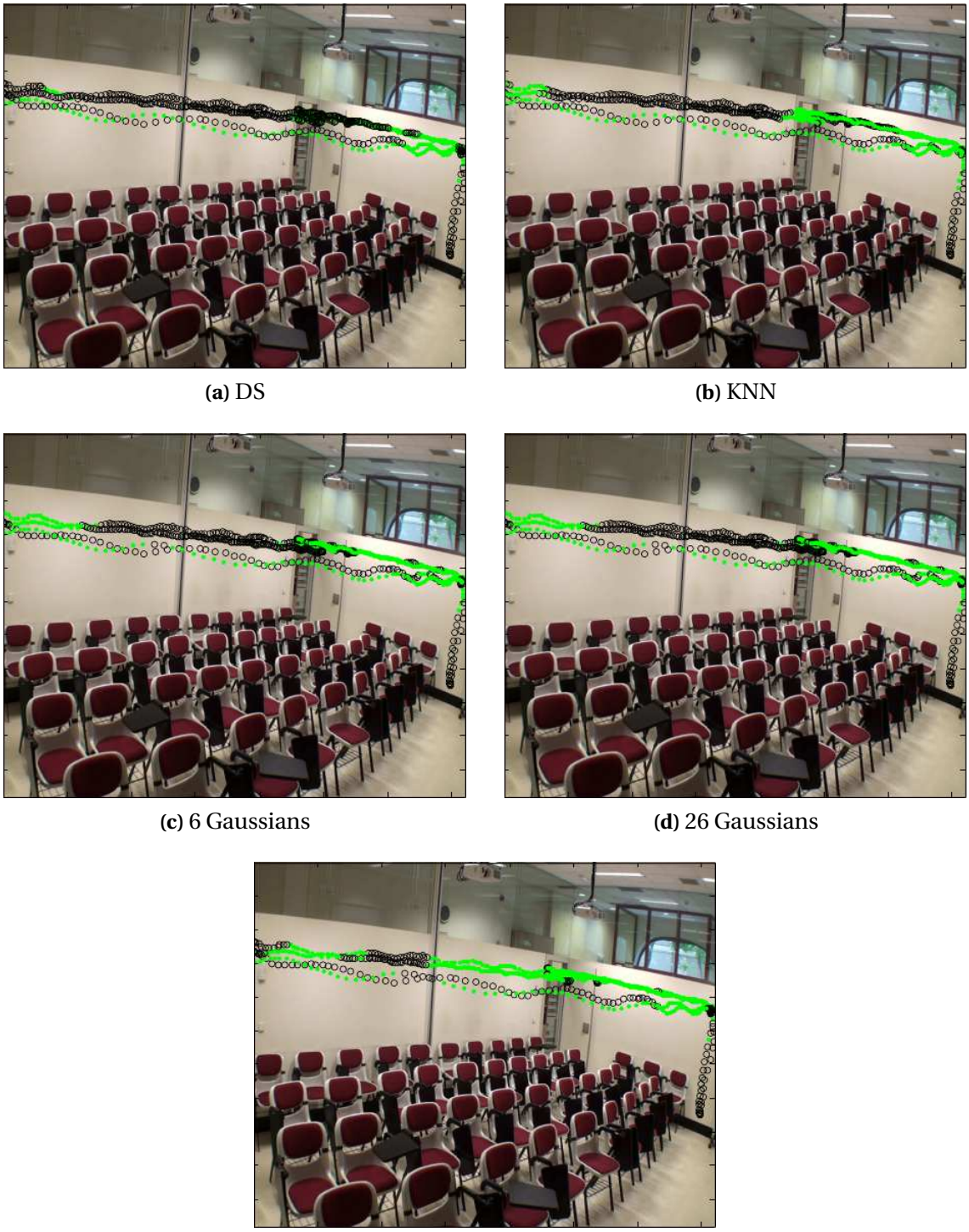


(e) FKM

Figure 4.15. Normal behavior test set detection results with all methods in IIT1 (case 4). Green dots are normal behaviors while black circles are abnormal detections.

abnormal. All the cases belong to limit points between normal behaviors and abnormal ones and the result would be that the alarm triggers a few seconds earlier or later than the optimal ground truth.

In cases 4 and 5 of the IIT video, the bigger part of FP error is situated within the



(a) DS **(b)** KNN

(c) 6 Gaussians **(d)** 26 Gaussians

(e) FKM

Figure 4.16. Abnormal behavior test set detection results with all methods in IIT1 (case 5). Green dots are normal behaviors while black circles are abnormal detections.

limits of the image. The people in the image disappear briefly, generating extraneous values of speed and direction causing a wrong classification. The FN is wrongly classified as known behavior in zones where the people changed direction in the training set. A person walking in the opposite direction to the normal samples is considered, in

(a) DS

(b) KNN

(c) 6 Gaussians

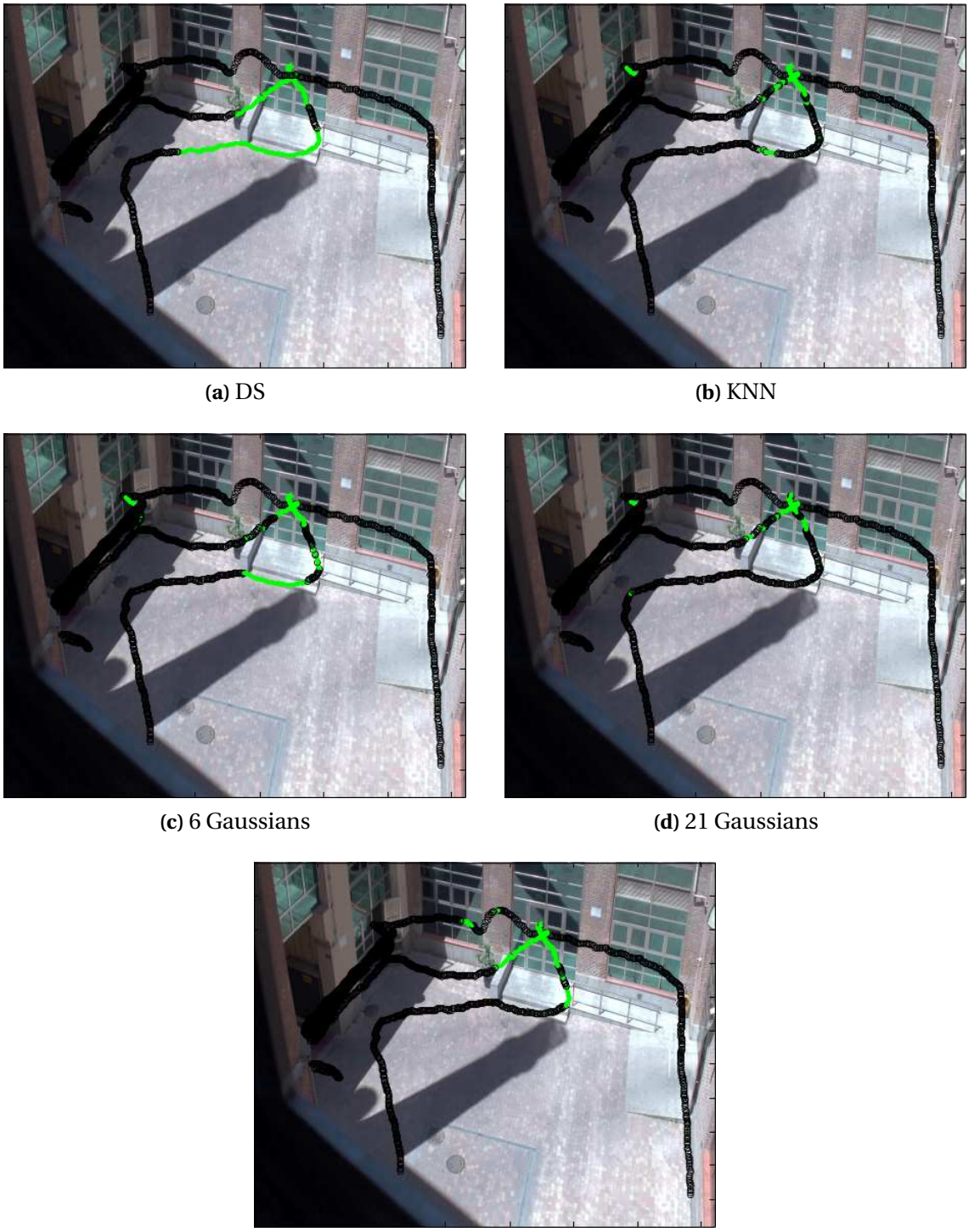
(d) 21 Gaussians

(e) FKM

Figure 4.17. Normal behavior test set detection results with all methods in ICAI1 (case 6). Green dots are normal behaviors while black circles are abnormal detections.

certain cases, to have a normal behavior near the right border. This is due to the fact that (x, y) locations of the abnormal trajectory near this border are similar to some of the coordinates of the normal trajectories used in the training step.

In the next two cases, the trajectory of a person zigzagging generates the high ratio of



(a) DS **(b)** KNN

(c) 6 Gaussians **(d)** 21 Gaussians

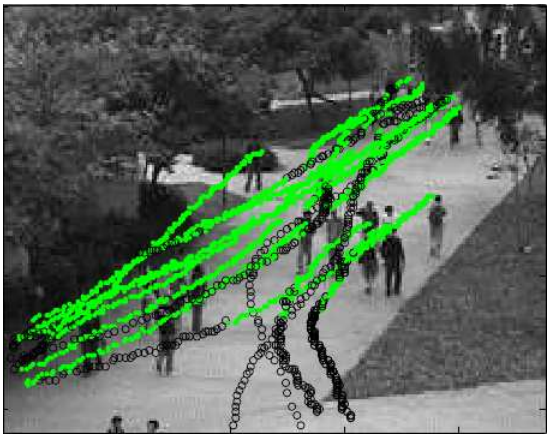
(e) FKM

Figure 4.18. Abnormal behavior test set detection results with all methods in ICAI1 (case 7). Green dots are normal behaviors while black circles are abnormal detections.

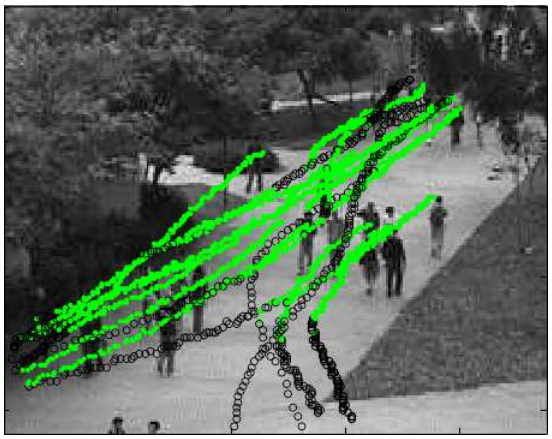
false-negative detections. Although a large part of the trajectory is classified as abnormal, some points are misclassified. They are considered to be similar to the trajectory of the training set of a person traversing the image from left to right.

Finally, DS mislabels several behaviors in the last case. On the one hand, it incorrectly

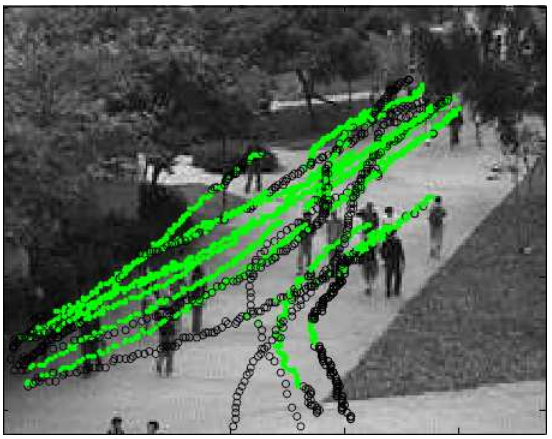
(a) DS



(b) KNN



(c) 7 Gaussians



(d) 30 Gaussians

(e) FKM

Figure 4.19. Abnormal behavior test set detection results with all methods in UCSD (case 8). Green dots are normal behaviors while black circles abnormal detections.

matches bikers and skaters with normal behaviors, because the speed difference is not big enough. On the other hand, it incorrectly labels as abnormal the people zigzagging to avoid other people walking at a different pace or in the opposite direction.

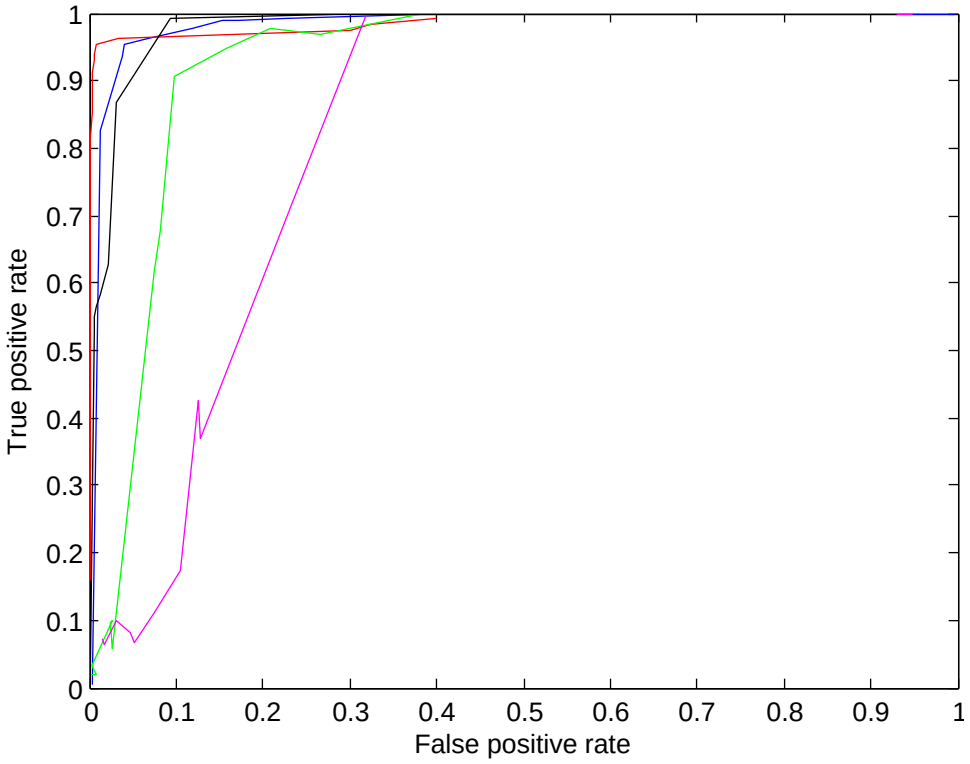


Figure 4.20. Results in ROC space, where blue, red, green, magenta and black curves correspond to DS, KNN, Op. MoG, DS MoG. and FKM results respectively in case 3.

4.6 Conclusions

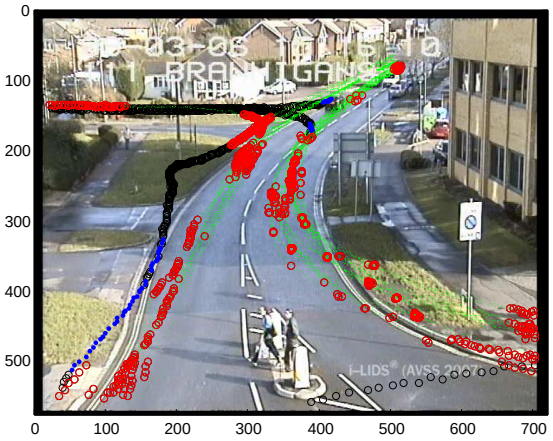
This chapter describes a solution to the video analysis problem of the automatic detection of abnormal events with the exclusive use of tracking algorithm output and no additional information about the environment. Dominant Sets, an unsupervised learning framework, is used to model normal behaviors which lead to an easier detection of anomalous behaviors.

The model proposed in this chapter is a viable solution for abnormal behavior detection. First, the data needed is easily obtainable from any usual object tracking system, without the need for any further information. Secondly, the training process is carried out automatically and user intervention is not necessary. These features make a surveillance system based on this model easily installable for any user.

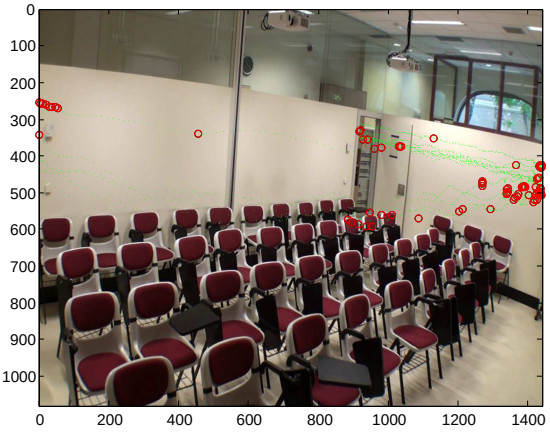
It has been experimentally proven that the method proposed herein has an overall better performance than the other usual clustering methods used for comparison. It has a stable functioning, where error rates remain low, except in two difficult cases where the other methods have even higher error rates. Moreover, the larger part of the errors made occurred in special situations. On the one hand, some errors took place within the limits of the image, where data could be wrongly collected. On the other hand, other errors were caused/provoked by a person who, a few moments earlier or later, acted strangely, and these errors were correctly detected by the system. Thus, in these cases

the problem would be minimal as the alarm would trigger anyway.

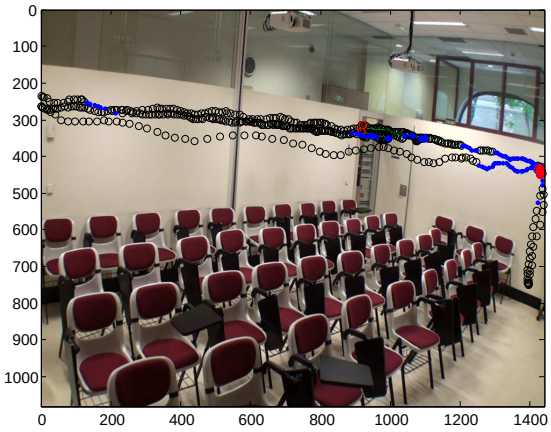
Dominant Sets is therefore a powerful and robust method to model behaviors and detect anomalies in scenes under surveillance.



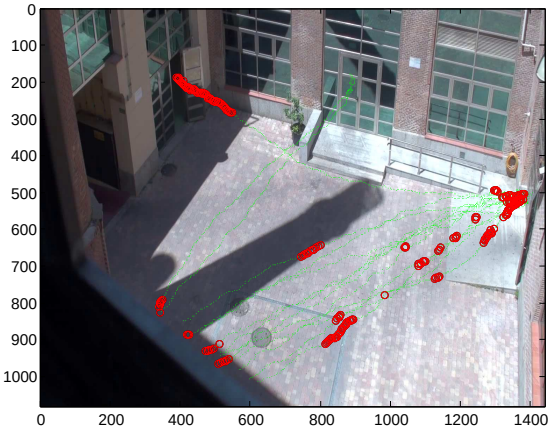
(a) Case 3



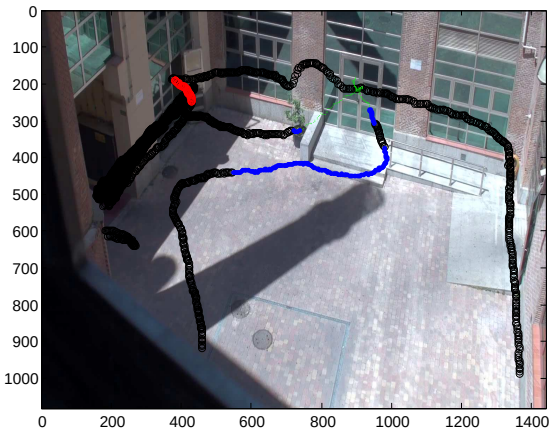
(b) Case 4



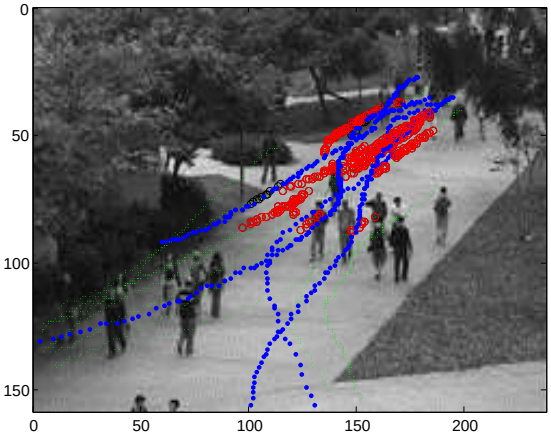
(c) Case 5



(d) Case 6



(e) Case 7



(f) Case 8

Figure 4.21. Extended results with DS. Blue points are FN and red circles are FP.

Chapter 5

Conclusions and future work

5.1 Conclusions

Security systems based on computer vision are increasingly common. Classical surveillance systems based on computer vision have been used for years, however, their utility remains very limited. Generally, they are used in a passive way. That means the system simply shows the scene to the operator at the same time as it is being recorded, but without analyzing the data and much less making a decision on how to react. The aim of these systems is simply to show the scene in the hope that the operator is aware and can take a decision if a dangerous situation arises, or as a resource to know what has happened *a posteriori*. Thus, the tendency is to create integrated smart security systems to cope with the necessities of the users. As such, image processing is becoming a promising and highly interesting research area.

This thesis sets forth a new methodology to cope with every problem a whole security system has to deal with. The three main steps developed solve the problems of detecting movement (see Chapter 2), tracking interesting objects (see Chapter 3) and deciding whether the detected behavior is anomalous or not and thus if an alarm should be activated (see Chapter 4).

The main constraint of a commercial system is the requirement to run in real time. In order to fulfill this requirement, in this thesis the motion detection and the object tracking modules proposed are strongly focused on optimization of time and resources, while maintaining the error rate low, allowing the behavior analysis module to be complex enough to deal with a wide number of scenarios.

First, a motion detection model called MMGA is proposed; MMGA uses the background modeling of MGM and the updating process of RTDENN, and is suitable in any surveillance scenario. The performance of the new updating process outperforms, with respect to time and resource consumption, the most used background subtraction techniques, while maintaining the error rate at the same level. The consumed time

reduction of MMGA, compared to MGM, attains up to 57%, while reliability increments by 2.5%. Furthermore, a second motion detection technique, called SDGM is presented. SDGM utilizes quite easily obtainable information about the environment: where the interesting objects can enter or exit the scene. With this information, it can restrict the search for foreground pixels to limited interesting regions, and subsequently modify them from frame to frame depending on the object's position and movement. SDGM can reduce the time needed for motion detection to a third of the time needed by other common background subtraction techniques. Thus SDGM is recommended in any case where data about entrances is available, and MMGA is recommended in the rest.

Secondly, an object tracker has been developed that is a low-resource consumer and easily deployed in any environment. It forms groups of objects tracked in previous frames with present detected blobs on a proximity basis. Subsequently, it matches blobs and objects belonging to the same group, taking into account the color distance between them. The locations each object visits are stored in segments. These segments allow the algorithm to tackle occlusions of interesting objects at the same time as it builds a set of possible trajectories, and calculates the probability of each one being the real trajectory, of each occluded object. It has been proven that the small errors that this system may incur in, under 1%, do not impede the creation of useful information for a correct abnormal behavior detection.

Thirdly, a new clusterization method has been applied to model normal behavior and to anomaly detection. Dominant Sets, applied to abnormal behavior detection, has proven to be a very effective and robust method compared to other classical clusterization methods. The mean error rate obtained by Dominant Sets is lower than 15% and generaly under 10%, whereas all the other clusterization techniques studied kept it higher than 20%. Moreover, unlike many other abnormal detection systems, Dominant Sets is an unsupervised clusterization model, making it a perfect fit for a commercial security system. It is important to be able to install the system without a great deal of technical knowledge to make it viable for and appealing to the end user. The abnormal detection system presented is a powerful and robust method that models behaviors and detects abnormalities in the scenes under surveillance.

5.2 Contributions

The development of this thesis has yielded several original contributions to the current knowledge about computer vision models. These are summarized below:

i. A new mathematical model describing *gates* has been designed. This model permits selection of the regions of interest in the scene, allowing search of movement to focus on those regions. Furthermore, the model is composed of a static part (static gates) and a dynamic one (dynamic gates). The former monitor the scene to

broaden the number of regions in the case that new interesting objects make an entry, while the latter is in charge of updating the search regions as it can efficiently modify them to follow the movement of the scene.

ii. The Static and Dynamic Gates Model develops the application of the gates mathematical model to the background subtraction problem. MGM, the most commonly used background subtraction algorithm, is used as the base algorithm to demonstrate the benefits of SDMG. A newly created updating process differentiates pixels belonging to a gate from those not belonging to any of them, in a way that takes advantage of the highlighted interesting regions of the image. The results obtained show the great reduction in time consumption of the SDGM compared to the MGM thanks to the focused motion detection analysis.

iii. The Mixture of Merged Gaussians Algorithm implements a novel application of the RTDNN algorithm in background subtraction algorithms. This model tackles the problem of time consumption in updating processes of the background subtraction techniques, a problem already raised by the SDGM. MMGA efficiently updates the background model, optimizing time consumption while keeping the error rate at the same level. New updating and segmentation steps are created to assimilate the complexity of RTDENN. These processes do not need any additional information about the environment and thus they are directly implementable in usual background subtraction techniques.

iv. An implementation of Dominant Sets for the detection of quick abnormal behaviors has been developed. The viability and undeniable interest for behavior analysis security system applications has been demonstrated, although Dominant Sets has never been applied to this field before. An unsupervised abnormal behavior detection model that only requires the data furnished by any usual object tracking system is proposed. The training of the model is carried out automatically without any user intervention, making it a perfect fit for an easy-to-install security system. The input training data set, as well as the data information needed to detect abnormal behaviors, used during the experiments were no more than the position of the tracked object during the last few frames. This information is easily collected in any tracking system. This model is thus easily implementable and a new interesting alternative for modern security systems.

Contributions i and ii were originally presented in [Alvar et al., 2014a]. [Alvar et al., 2013] detailed contribution iii. Finally, [Alvar et al., 2014b] developed contribution iv.

In addition to these main contributions, this thesis has produced diverse improvements listed below:

i. The presentation of a new error measurement system called “extra error”. The aim of this new measurement is to test how the error varies from one background subtraction technique to another. It is especially useful when the main improvement

of a novel technique is not the reduction of the error rate, as was the case in both SDGM and MMGA, which focused on time reduction.

ii. The division of the different cases a tracking algorithm must deal with into six smaller problems, as well as the solution of these cases. This division permits the tracking system to save up unnecessary computing time to solve straightforward problems, while it robustly tracks when more complicated situations occur, namely *case 6s*.

iii. The development of a new probabilistic model based on *segments*. The tracking system is an essential step in people behavior analysis systems, but common current tracking systems struggle with complicated scenes that have several occlusions. This new probabilistic tracking model copes with occlusion problems. Each occluded object is modeled in a way that may have different possible trajectories with different probabilities, updated in real time, depending on the information gathered.

iv. To correctly test the different models proposed in this thesis work, ground truth data of many videos was produced. In Chapter 2, the background/foreground segmentation ground truth was manually obtained for videos 1 to 4. No tracking information was publicly available for any video, therefore this information was created in order to provide a comparison tool. In the same way, ground truth normal/abnormal behavior was manually tagged to constitute a comparison tool between models. Furthermore, two new videos, $IIT1$ and $ICAI1$, were created to widen the comparison videos and demonstrate the performance of the abnormal behavior detection models with new people behaviors. As has been pointed out, all of the new data created is available on the author´s webpage[1].

5.3 Further developments

This thesis presents a whole security system aimed at autonomously detecting abnormal behaviors in a scene. However, some improvements may be added to reinforce and enhance it.

5.3.1 SDGM and MMGA merger

This thesis proposes two different background subtraction techniques, SDGM and MMGA. Section 2.7.1 showed that SDGM can be implemented at the same time as other MGM improvements. The extra error analysis carried out demonstrated that

[1] http://www.iit.upcomillas.es/malvar/

SDGM does not incur in further error compared to MGM. This means that any model improving MGM can, theoretically, be applied in the same way to SDGM.

A new model unifying the virtues of both methods may thus be developed. This new model merging SDGM and MMGA would restrict the motion detection to some interesting areas, in a similar way to SDGM. Moreover, the updating process would be a new application of the RTDENN updating, evolving from the one proposed for MMGA. Theoretically, SDGMMGA should, at least, reduce the time needed for its execution to a level similar to that of other techniques, while, obviously, maintaining the error rate sufficiently low.

5.3.2 Choice of automatic behavior components

During the tracking step, a large amount of data is gathered for each object. The clusterization method chosen to model the interesting objects' behavior, Dominant Sets, can work in much bigger spaces, although this penalizes its computing speed. Nevertheless, during the clusterization step, only the present position, speed and direction are used. Position gives present information while speed and direction complement it with the information about past positions. These components are those most commonly used in behavior clusterization, and have indeed proven to be a good choice in order to obtain robust results.

However, there are still numerous components available in the algorithm, without changing the tracking step, to cluster the data (position, speed and direction of all the previous points of the same trajectory). Dominant Sets can be also used to optimally choose the best components for an efficient clusterization of the training data set. That way, only the components carrying the most meaningful information are kept, avoiding unnecessary dependencies between them.

The first test carried out with small computer-generated data sets shows promising results in this field. A totally autonomous system based on Dominant Sets capable of choosing the best axis throughout the whole of the available components is thus possible, in a similar way to PCA. Moreover, as this analysis can be carried out during the system training step, it does not have such a huge time restriction. Finally, it could allow choice of the axis that perfectly fits each application, without any user operation.

Bibliography

[Adam et al., 2008] Adam, A., Rivlin, E., Shimshoni, I., and Reinitz, D. (2008). Robust real-time unusual event detection using multiple fixed-location monitors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(3):555–560.

[Aggarwal and Cai, 1999] Aggarwal, J. and Cai, Q. (1999). Human motion analysis: A review. *Computer Vision and Image Understanding*, 73(3):428 – 440.

[Aggarwal and Ryoo, 2011] Aggarwal, J. and Ryoo, M. (2011). Human activity analysis: A review. *ACM Comput. Surv.*, 43(3):16:1–16:43.

[Albarelli et al., 2009] Albarelli, A., Bulò, S. R., Torsello, A., and Pelillo, M. (2009). Matching as a non-cooperative game. In *Computer Vision, IEEE 12th International Conference on*, pages 1319–1326.

[Albarelli et al., 2010] Albarelli, A., Rodolà, E., Cavallarin, A., and Torsello, A. (2010). Robust figure extraction on textured background: A game-theoretic approach. In *Pattern Recognition, 20th International Conference on*, pages 360–363.

[Albarelli et al., 2012] Albarelli, A., Rodolà, E., and Torsello, A. (2012). Imposing semi-local geometric constraints for accurate correspondences selection in structure from motion: A game-theoretic perspective. *International Journal of Computer Vision*, 97:36–53.

[Ali and Shah, 2007] Ali, S. and Shah, M. (2007). A lagrangian particle dynamics approach for crowd flow segmentation and stability analysis. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 1–6.

[Alvar et al., 2013] Alvar, M., Rodriguez-Calvo, A., Sanchez-Miralles, A., and Arranz, A. (2013). Mixture of merged gaussian algorithm using rtdenn. *Machine Vision and Applications*, pages 1–12.

[Alvar et al., 2014a] Alvar, M., Sánchez, A., and Arranz, A. (2014a). Fast background subtraction using static and dynamic gates. *Artificial Intelligence Review*, 41(1):113–128.

[Alvar et al., 2014b] Alvar, M., Torsello, A., Sanchez-Miralles, A., and Armingol, J.-M. (2014b). Abnormal behavior detection using dominant sets. *Machine Vision and Applications*, pages 1–18.

[Antic and Ommer, 2011] Antic, B. and Ommer, B. (2011). Video parsing for abnormality detection. In *Computer Vision, IEEE International Conference on*, pages 2415–2422.

[Barron et al., 1992] Barron, J., Fleet, D., Beauchemin, S., and Burkitt, T. A. (1992). Performance of optical flow techniques. In *Computer Vision and Pattern Recognition. Proceedings of IEEE Computer Society Conference on*, pages 236–242.

[Barron et al., 1994] Barron, J. L., Fleet, D. J., and Beauchemin, S. S. (1994). Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77.

[Bay et al., 2008] Bay, H., Ess, A., Tuytelaars, T., and Gool, L. V. (2008). Speeded-up robust features (surf). *Computer Vision and Image Understanding*, 110(3):346 – 359. Similarity Matching in Computer Vision and Multimedia.

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Gool, L. (2006). Surf: Speeded up robust features. In Leonardis, A., Bischof, H., and Pinz, A., editors, *European Conference on Computer Vision - ECCV*, volume 3951 of *Lecture Notes in Computer Science*, pages 404–417. Springer Berlin Heidelberg.

[Beauchemin and Barron,] Beauchemin, S. S. and Barron, J. L. The computation of optical flow. *ACM Computing Surveys*.

[Beleznai et al., 2007] Beleznai, C., Sommer, P., and Bischof, H. (2007). Scale-adaptive clustering for object detecion and counting. In *IEEE International Workshop on PETS*, pages 9–16.

[Benezeth et al., 2009] Benezeth, Y., Jodoin, P.-M., Saligrama, V., and Rosenberger, C. (2009). Abnormal events detection based on spatio-temporal co-occurences. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 2458–2465.

[Boal et al., 2014] Boal, J., Sï¿½nchez-Miralles, l., and Arranz, l. (2014). Topological simultaneous localization and mapping: a survey. *Robotica*, FirstView:1–19.

[Bobick and Davis, 2001] Bobick, A. and Davis, J. (2001). The recognition of human movement using temporal templates. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(3):257 –267.

[Bobick and Wilson, 1997] Bobick, A. and Wilson, A. (1997). A state-based approach to the representation and recognition of gesture. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 19(12):1325 –1337.

[Bowyer et al., 1999] Bowyer, K., Kranenburg, C., and Dougherty, S. (1999). Edge detector evaluation using empirical roc curves. In *Computer Vision and Pattern Recognition. Computer Society Conference on*, volume 1, pages 354–359.

[Bowyer et al., 2001] Bowyer, K., Kranenburg, C., and Dougherty, S. (2001). Edge detector evaluation using empirical roc curves. *Computer Vision and Image Understanding*, 84(1):77 – 103.

[Brand et al., 1997] Brand, M., Oliver, N., and Pentland, A. (1997). Coupled hidden markov models for complex action recognition. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, pages 994 – 999.

[Bregler, 1997] Bregler, C. (1997). Learning and recognizing human dynamics in video sequences. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, pages 568 – 574.

[Brémond and Medioni, 1998] Brémond, F. and Medioni, G. (1998). Scenario recognition in airborne video imagery. In *Interpretation of Visual Motion Workshop, Computer Vision and Pattern Recognition*, pages 57 – 64.

[Brémond et al., 2006] Brémond, F., Thonnat, M., and Zúñiga, M. (2006). Video-understanding framework for automatic behavior recognition. *Behavior Research Methods*, 38(3):416–426.

[Brown et al., 2005] Brown, L., Senior, A., Tian, Y., Connell, J., Hampapur, A., Shu, C., Merkl, H., and Lu, M. (2005). Performance evaluation of surveillance systems under varying conditions. In *IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pages 79–87.

[Buxton and Mukerjee, 2000] Buxton, H. and Mukerjee, A. (2000). Conceptualizing images. *Image and Vision Computing*, 18(2):79 – 79.

[Cannons, 2008] Cannons, K. (2008). A review of visual tracking. Technical Report CSE-2008-07, Department of Computer Science and Engineering, York University, Retrieved from http://www.eecs.yorku.ca/research/techreports/2008/CSE-2008-07.pdf.

[Canny, 1986] Canny, J. (1986). A computational approach to edge detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 8(6):679–698.

[Cermak and Keyzer, 2007] Cermak, G. and Keyzer, K. (2007). Unsupervised intrusion detection using color images. In Bebis, G., Boyle, R., Parvin, B., Koracin, D., Paragios, N., Tanveer, S.-M., Ju, T., Liu, Z., Coquillart, S., Cruz-Neira, C., Müller, T., and Malzbender, T., editors, *Advances in Visual Computing*, volume 4842 of *Lecture Notes in Computer Science*, pages 770–780. Springer Berlin Heidelberg.

[Chen et al., 2012] Chen, S., Zhang, J., Li, Y., and Zhang, J. (2012). A hierarchical model incorporating segmented regions and pixel descriptors for video background subtraction. *Industrial Informatics, IEEE Transactions on*, 8(1):118 –127.

[Chomat and Crowley, 1998] Chomat, O. and Crowley, J. L. (1998). Recognizing motion using local appearance. *Journal of Intelligent and Robotic Systems*, pages 271–279.

[Comaniciu and Meer, 2002] Comaniciu, D. and Meer, P. (2002). Mean shift: a robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(5):603–619.

[Comaniciu et al., 2001] Comaniciu, D., Ramesh, V., and Meer, P. (2001). The variable bandwidth mean shift and data-driven scale selection. In *Computer Vision. Proceedings. Eighth IEEE International Conference on*, volume 1, pages 438–445.

[Comaniciu et al., 2003] Comaniciu, D., Ramesh, V., and Meer, P. (2003). Kernel-based object tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(5):564–577.

[Cong et al., 2011] Cong, Y., Yuan, J., and Liu, J. (2011). Sparse reconstruction cost for abnormal event detection. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 3449–3456.

[Cong et al., 2013] Cong, Y., Yuan, J., and Liu, J. (2013). Abnormal event detection in crowded scenes using sparse representation. *Pattern Recognition*, 46(7):1851 – 1864.

[Connell et al., 2004] Connell, J., Senior, A., Hampapur, A., Tian, Y.-L., Brown, L., and Pankanti, S. (2004). Detection and tracking in the ibm peoplevision system. In *Multimedia and Expo, IEEE International Conference on*, volume 2, pages 1403 – 1406.

[Cremers, 2006] Cremers, D. (2006). Dynamical statistical shape priors for level set-based tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(8):1262–1273.

[Cremers and Schnörr, 2003] Cremers, D. and Schnörr, C. (2003). Statistical shape knowledge in variational motion segmentation. *Image and Vision Computing*, 21(1):77 – 86.

[Cristani et al., 2002] Cristani, M., Bicego, M., and Murino, V. (2002). Integrated region- and pixel-based approach to background modelling. In *Proceedings of the Workshop on Motion and Video Computing*, pages 3–8.

[Cristani et al., 2010] Cristani, M., Farenzena, M., Bloisi, D., and Murino, V. (2010). Background subtraction for automated multisensor surveillance: A comprehensive review. *EURASIP Journal on Advances in Signal Processing*, 2010(1):1–24.

[Cucchiara et al., 2003] Cucchiara, R., Grana, C., Piccardi, M., and Prati, A. (2003). Detecting moving objects, ghosts, and shadows in video streams. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1337–1342.

[Dee and Hogg, 2004a] Dee, H. and Hogg, D. (2004a). Detecting inexplicable behaviour. In *Proceedings of the British Machine Vision Conference*, pages 50.1–50.10. BMVA Press.

[Dee and Hogg, 2004b] Dee, H. and Hogg, D. C. (2004b). Is it interesting? comparing human and machine judgements on the pets dataset. *Sixth International Workshop on Performance Evaluation of Tracking And Surveillance*, 33(1):49–55.

[Del Rose and Wagner, 2012] Del Rose, M. S. and Wagner, C. C. (2012). Survey on classifying human actions through visual sensors. *Artificial Intelligence Review*, 37(4):301–311.

[Dick and Brooks, 2003] Dick, A. R. and Brooks, M. J. (2003). Issues in automated visual surveillance. In *Digital Image Computing: Techniques and Applications. Proceedings of the VIIth Conference on*, volume 1, pages 195 – 204.

[Dockstader and Tekalp, 2000] Dockstader, S. L. and Tekalp, A. M. (2000). Real-time object tracking and human face detection in cluttered scenes. In *Image and Video Communications and Processing. Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE)*, volume 3974, pages 957–968.

[Elgammal et al., 2000] Elgammal, A., Harwood, D., and Davis, L. (2000). Non-parametric model for background subtraction. *Lecture Notes in Computer Science*, 1843:751–767.

[Feldman and Weinshall, 2008] Feldman, D. and Weinshall, D. (2008). Motion segmentation and depth ordering using an occlusion detector. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(7):1171–1185.

[Fieguth and Terzopoulos, 1997] Fieguth, P. and Terzopoulos, D. (1997). Color-based tracking of heads and other mobile objects at video frame rates. In *Computer Vision and Pattern Recognition. Proceedings of IEEE Computer Society Conference on*, pages 21–27.

[Fleet and Weiss, 2006] Fleet, D. and Weiss, Y. (2006). Optical flow estimation. In *Handbook of Mathematical Models in Computer Vision*, pages 237–257.

[Forsyth et al., 2005] Forsyth, D. A., Arikan, O., Ikemoto, L., O'Brien, J., and Ramanan, D. (2005). Computational studies of human motion: part 1, tracking and motion synthesis. *Found. Trends. Comput. Graph. Vis.*, 1(2-3):77–254.

[Gao et al., 2000] Gao, X., Boult, T., Coetzee, F., and Ramesh, V. (2000). Error analysis of background adaption. In *Computer Vision and Pattern Recognition. Proceedings. IEEE Conference on*, volume 1, pages 503–510 vol.1.

[Gavrila, 1999] Gavrila, D. M. (1999). The visual analysis of human movement: A survey. *Computer Vision and Image Understanding*, 73(1):82 – 98.

[Geronimo et al., 2010] Geronimo, D., Lopez, A., Sappa, A., and Graf, T. (2010). Survey of pedestrian detection for advanced driver assistance systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(7):1239–1258.

[Gong et al., 2011] Gong, S., Loy, C., and Xiang, T. (2011). Security and surveillance. In *Visual Analysis of Humans*, pages 455–472.

[Goyette et al., 2012] Goyette, N., Jodoin, P., Porikli, F., Konrad, J., and Ishwar, P. (2012). Changedetection.net: A new change detection benchmark dataset. In *Computer Vision and Pattern Recognition Workshops (CVPRW), IEEE Computer Society Conference on*, pages 1–8.

[Guo et al., 1994a] Guo, Y., Xu, G., and Tsuji, S. (1994a). Tracking human body motion based on a stick figure model. *Journal of Visual Communication and Image Representation*, 5:1–9.

[Guo et al., 1994b] Guo, Y., Xu, G., and Tsuji, S. (1994b). Understanding human motion patterns. In *Pattern Recognition - Conference B: Computer Vision & Image Processing., Proceedings of the 12th IAPR International. Conference on*, volume 2, pages 325–329.

[Hamid et al., 2005] Hamid, R., Johnson, A., Batta, S., Bobick, A., Isbell, C., and Coleman, G. (2005). Detection and explanation of anomalous activities: representing activities as bags of event n-grams. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages 1031–1038.

[Hamid et al., 2009] Hamid, R., Maddi, S., Johnson, A., Bobick, A., Essa, I., and Isbell, C. (2009). A novel sequence representation for unsupervised analysis of human activities. *Artificial Intelligence*, 173(14):1221–1244.

[Hampapur et al., 2003] Hampapur, A., Brown, L., Connell, J., Pankanti, S., Senior, A., and Tian, Y. (2003). Smart surveillance: applications, technologies and implications. In *Joint Conference of the Fourth International Conference on Information, Communications and Signal Processing, and the Fourth Pacific Rim Conference on Multimedia.*, volume 2, pages 1133 – 1138.

[Haritaoglu et al., 2000] Haritaoglu, I., Harwood, D., and Davis, L. (2000). W4: real-time surveillance of people and their activities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):809–830.

[Heikkila and Pietikainen, 2006] Heikkila, M. and Pietikainen, M. (2006). A texture-based method for modeling the background and detecting moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(4):657–662.

[Horprasert et al., 1999] Horprasert, T., Harwood, D., and Davis, L. S. (1999). Computer vision, frame rate workshop. proceedings of the 7th ieee international conference on. In *A statistical approach for real-time robust background subtraction and shadow detection*, volume 4, pages 1 – 19.

[Howarth, 2005] Howarth, R. J. (2005). Spatial models for wide-area visual surveillance: Computational approaches and spatial building-blocks. *Artificial Intelligence Review*, 23(2):97–155.

[Hu et al., 2004a] Hu, W., Tan, T., Wang, L., and Maybank, S. (2004a). A survey on visual surveillance of object motion and behaviors. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 34(3):334–352.

[Hu et al., 2006] Hu, W., Xiao, X., Fu, Z., Xie, D., Tan, T., and Maybank, S. (2006). A system for learning statistical motion patterns. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(9):1450 –1464.

[Hu et al., 2004b] Hu, W., Xie, D., Tan, T., and Maybank, S. (2004b). Learning activity patterns using fuzzy self-organizing neural network. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 34(3):1618 –1626.

[Hung and Kröse, 2011] Hung, H. and Kröse, B. (2011). Detecting f-formations as dominant sets. In *Proceedings of the 13th international conference on multimodal interfaces*, pages 231–238, New York, NY, USA. ACM.

[Jain and Dubes, 1988] Jain, A. K. and Dubes, R. C. (1988). *Algorithms for clustering data*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[Jan, 2004] Jan, T. (2004). Neural network based threat assessment for automated visual surveillance. In *Neural Networks. Proceedings. IEEE International Joint Conference on*, volume 2, pages 1309 – 1312.

[Jiang et al., 2011] Jiang, F., Yuan, J., Tsaftaris, S. A., and Katsaggelos, A. K. (2011). Anomalous video event detection using spatiotemporal context. *Computer Vision and Image Understanding*, 115(3):323 – 333.

[Johnson and Hogg, 1996] Johnson, N. and Hogg, D. (1996). Learning the distribution of object trajectories for event recognition. *Image and Vision Computing*, 14(8):609 – 615.

[Kailath, 1967] Kailath, T. (1967). The divergence and bhattacharyya distance measures in signal selection. *Communication Technology, IEEE Transactions on*, 15(1):52–60.

[Kaltsa et al., 2012] Kaltsa, V., Briassouli, A., Kompatsiaris, I., and Strintzis, M. (2012). Timely, robust crowd event characterization. In *Image Processing, 19th IEEE International Conference on*, pages 2697–2700.

[Khan et al., 2011] Khan, Z., Gu, I.-H., and Backhouse, A. (2011). Robust visual object tracking using multi-mode anisotropic mean shift and particle filters. *Circuits and Systems for Video Technology, IEEE Transactions on*, 21(1):74–87.

[Kim and Park, 2006] Kim, B.-G. and Park, D.-J. (2006). Novel target segmentation and tracking based on fuzzy membership distribution for vision-based target tracking system. *Image and Video Computing*, 24(12):1319–1331.

[Kim and Grauman, 2009] Kim, J. and Grauman, K. (2009). Observe locally, infer globally: A space-time mrf for detecting abnormal activities with incremental updates. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 2921–2928.

[Kim et al., 2005] Kim, K., Chalidabhongse, T. H., Harwood, D., and Davis, L. (2005). Real-time foreground-background segmentation using codebook model. *Real-Time Imaging*, 11(3):172–185.

[Ko, 2008] Ko, T. (2008). A survey on behavior analysis in video surveillance for homeland security applications. In *37th IEEE Applied Imagery Pattern Recognition Workshop*, pages 1 –8.

[Kojima et al., 2000] Kojima, A., Izumi, M., Tamura, T., and Fukunaga, K. (2000). Generating natural language description of human behavior from video images. In *Pattern Recognition. 15th International Conference on*, volume 4, pages 728–731.

[Kojima et al., 2002] Kojima, A., Tamura, T., and Fukunaga, K. (2002). Natural language description of human activities from video images based on concept hierarchy of actions. *International Journal of Computer Vision*, 50:171–184.

[Kollnig et al., 1994] Kollnig, H., Nagel, H.-H., and Otte, M. (1994). Association of motion verbs with vehicle movements extracted from dense optical flow fields. In *Proceedings of the Third European Conference-Volume II on Computer Vision*, pages 338–347, London, UK. Springer-Verlag.

[Lee, 2005] Lee, D.-S. (2005). Effective gaussian mixture learning for video background subtraction. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 27(5):827–832.

[Leichter et al., 2010] Leichter, I., Lindenbaum, M., and Rivlin, E. (2010). Mean shift tracking with multiple reference color histograms. *Computer Vision and Image Understanding*, 114(3):400 – 408.

[Li et al., 2010] Li, C.-L., Hao, Z.-B., and Li, J.-J. (2010). Abnormal behavior detection using a novel behavior representation. In *Apperceiving Computing and Intelligence Analysis, International Conference on*, pages 331–336.

[Li and Wang, 1993] Li, H. and Wang, J. (1993). Computing optical flow with a recurrent neural network. *Journal of Pattern Analysis and Artificial Intelligence*, 7:801–814.

[Li et al., 2004] Li, L., Huang, W., Gu, I. Y.-H., and Tian, Q. (2004). Statistical modeling of complex backgrounds for foreground object detection. *Image Processing, IEEE Transactions on*, 13(11):1459 –1472.

[Li et al., 2014] Li, W., Mahadevan, V., and Vasconcelos, N. (2014). Anomaly detection and localization in crowded scenes. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36(1):18–32.

[Li et al., 2013] Li, X., Hu, W., Shen, C., Zhang, Z., Dick, A., and Hengel, A. V. D. (2013). A survey of appearance models in visual object tracking. *ACM Transactions on Intelligent Systems and Technology*, Accepted:1 – 41.

[Li et al., 2008] Li, Z., Tang, Q., and Sang, N. (2008). Improved mean shift algorithm for occlusion pedestrian tracking. *Electronics Letters*, 44(10):622–623.

[Lin et al., 1999] Lin, C.-T., Nein, H.-W., and Lin, W.-C. (1999). A space-time delay neural network for motion recognition and its application to lipreading. *International journal of neural systems*, 9(4):311–334.

[Lipton et al., 1998] Lipton, A. J., Fujiyoshi, H., and Patil, R. S. (1998). Moving target classification and tracking from real-time video. In *Applications of Computer Vision. Proceedings of the Fourth IEEE Workshop on*, pages 8–14.

[Liu et al., 1998] Liu, H., Hong, T.-H., Herman, M., Camus, T., and Chellappa, R. (1998). Accuracy vs efficiency trade-offs in optical flow algorithms. *Computer Vision and Image Understanding*, 72(3):271 – 286.

[Liu et al., 2012] Liu, L., Sang, N., and Yang, S. (2012). A low-cost hand gesture human-computer interaction system. In *Consumer Electronics, IEEE International Conference on*, pages 299–300.

[Liu et al., 2005] Liu, X., Tu, P., Rittscher, J., Perera, A., and Krahnstoever, N. (2005). Detecting and counting people in surveillance applications. In *Advanced Video and Signal Based Surveillance. Conference on*, pages 306 – 311.

[Lou et al., 2002] Lou, J., Liu, Q., Tan, T., and Hu, W. (2002). Semantic interpretation of object activities in a surveillance system. In *Pattern Recognition. Proceedings. 16th International Conference on*, volume 3, pages 777 – 780.

[Lowe, 2004] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60(2):91–110.

[Lucas and Kanade, 1981] Lucas, B. and Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. In *International Joint Conference in Artificial Intelligence*, pages 674–679.

[Maddalena and Petrosino, 2008] Maddalena, L. and Petrosino, A. (2008). A self-organizing approach to background subtraction for visual surveillance applications. *Image Processing, IEEE Transactions on*, 17(7):1168 –1177.

[Mahadevan et al., 2010] Mahadevan, V., Li, W., Bhalodia, V., and Vasconcelos, N. (2010). Anomaly detection in crowded scenes. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 1975–1981.

[McKenna et al., 2000] McKenna, S. J., Jabri, S., Duric, Z., Rosenfeld, A., and Wechsler, H. (2000). Tracking groups of people. *Computer Vision and Image Understanding*, 80(1):42 – 56.

[Mehran et al., 2009] Mehran, R., Oyama, A., and Shah, M. (2009). Abnormal crowd behavior detection using social force model. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 935–942.

[Meier et al., 2000] Meier, U., Stiefelhagen, R., Yang, J., and Waibel, A. (2000). Towards unrestricted lip reading. *International Journal of Pattern Recognition and Artificial Intelligence*, 14(5):571–585.

[Moeslund and Granum, 2001] Moeslund, T. B. and Granum, E. (2001). A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding*, 81(3):231 – 268.

[Munkelt and Kirchner, 1996] Munkelt, O. and Kirchner, H. (1996). Stabil: A system for monitoring persons in image sequences. In *Image and Video Processing. Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE)*, volume 2666, pages 163–179.

[Myers et al., 1980] Myers, C., Rabiner, L., and Rosenberg, A. (1980). Performance tradeoffs in dynamic time warping algorithms for isolated word recognition. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 28(6):623 – 635.

[Nayak et al., 2011] Nayak, N. M., Sethi, R. J., Song, B., and Roy-Chowdhury, A. K. (2011). Modeling and recognition of complex human activities. In Moeslund, T. B., Hilton, A., Kruger, V., and Sigal, L., editors, *Visual Analysis of Humans*, pages 289–309. Springer London.

[Nummiaro et al., 2003] Nummiaro, K., Koller-Meier, E., and Gool, L. V. (2003). An adaptive color-based particle filter. *Image and Vision Computing*, 21(1):99–110.

[Ohta et al., 1980] Ohta, Y.-I., Kanade, T., and Sakai, T. (1980). Color information for region segmentation. *Computer Graphics and Image Processing*, 13(3):222–241.

[Oliver et al., 2000] Oliver, N. M., Rosario, B., and Pentland, A. P. (2000). A bayesian computer vision system for modeling human interactions. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):831–843.

[Owens and Hunter, 2000] Owens, J. and Hunter, A. (2000). Application of the self-organising map to trajectory classification. In *Visual Surveillance. Proceedings. Third IEEE International Workshop on*, pages 77 –83.

[Paragios and Deriche, 2000] Paragios, N. and Deriche, R. (2000). Geodesic active contours and level sets for the detection and tracking of moving objects. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(3):266–280.

[Paschos, 2001] Paschos, G. (2001). Perceptually uniform color spaces for color texture analysis: an empirical evaluation. *Image Processing, IEEE Transactions on*, 10(6):932–937.

[Pavan and Pelillo, 2004] Pavan, M. and Pelillo, M. (2004). Efficient out-of-sample extension of dominant-set clusters. In *Advances in Neural Information Processing Systems 17*, pages 1057–1064.

[Pavan and Pelillo, 2007] Pavan, M. and Pelillo, M. (2007). Dominant sets and pairwise clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 29(1):167 –172.

[Pelillo, 2009] Pelillo, M. (2009). What is a cluster? perspectives from game theory. In *NIPS Workshop on Clustering: Science of Art*.

[Phillips et al., 2002] Phillips, P., Sarkar, S., Robledo, I., Grother, P., and Bowyer, K. (2002). The gait identification challenge problem: data sets and baseline algorithm. In *Pattern Recognition, 2002. Proceedings. 16th International Conference on*, volume 1, pages 385 – 388.

[Piccardi, 2004] Piccardi, M. (2004). Background subtraction techniques: a review. In *Systems, Man and Cybernetics. IEEE International Conference on*, volume 4, pages 3099 – 3104.

[Piccardi and Jan, 2004] Piccardi, M. and Jan, T. (2004). Mean-shift background image modelling. In *Image Processing. International Conference on*, volume 5, pages 3399–3402.

[Poppe et al., 2007] Poppe, C., Martens, G., Lambert, P., and Van de Walle, R. (2007). Improved background mixture models for video surveillance applications. *Lecture Notes in Computer Science*, 4843:251.

[Poritz, 1988] Poritz, A. (1988). Hidden markov models: a guided tour. In *Acoustics, Speech, and Signal Processing. International Conference on*, pages 7 –13 vol.1.

[Power and Schoonees, 2002] Power, P. and Schoonees, J. (2002). Understanding background mixture models for foreground segmentation. In *Proceedings Image and Vision Computing New Zealand*, pages 267–271.

[Regazzoni et al., 2001] Regazzoni, C., Ramesh, V., and Foresti, G. L. (2001). Special issue on video communications, processing, and understanding for third generation surveillance systems. *Proceedings of the IEEE*, 89(10):1355–1539.

[Remagnino et al., 1998a] Remagnino, P., Tan, T., and Baker, K. (1998a). Agent orientated annotation in model based visual surveillance. In *Computer Vision. Sixth International Conference on*, pages 857–862.

[Remagnino et al., 1998b] Remagnino, P., Tan, T., and Baker, K. (1998b). Multi-agent visual surveillance of dynamic scenes. *Image and Vision Computing*, 16(8):529–532.

[Remagnino et al., 1997] Remagnino, P. R., Baumberg, A., Grove, T., Hogg, D., Tan, T. N., Worrall, A. D., and Baker, K. D. (1997). An integrated traffic and pedestrian model_based vision system. In *British Machine Vision Conference*, pages 380–389.

[Rota Bulò and Bomze, 2011] Rota Bulò, S. and Bomze, I. M. (2011). Infection and immunization: A new class of evolutionary game dynamics. *Games and Economic Behavior*, 71(1):193–211.

[Saligrama and Chen, 2012] Saligrama, V. and Chen, Z. (2012). Video anomaly detection based on local statistical aggregates. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 2112–2119.

[Sánchez Miralles and Sanz Bobi, 2002] Sánchez Miralles, A. and Sanz Bobi, M. A. (2002). Real time dynamic ellipsoidal neural network (rtdenn). In *International Conference on Signal Processing, Robotics and Automation*, pages 1991–1995.

[Sánchez Miralles and Sanz Bobi, 2004] Sánchez Miralles, A. and Sanz Bobi, M. A. (2004). Global path planning in gaussian probabilistic maps. *Journal of Intelligent & Robotic Systems*, 40(1):89–102. 10.1023/B:JINT.0000034339.13257.e6.

[Sánchez Miralles and Sanz Bobi, 2006] Sánchez Miralles, A. and Sanz Bobi, M. A. (2006). A neural-based model for fast continuous and global robot location. *Journal of Intelligent & Robotic Systems*, 46(3):221–243. 10.1007/s10846-006-9046-4.

[Santner et al., 2010] Santner, J., Leistner, C., Saffari, A., Pock, T., and Bischof, H. (2010). Prost: Parallel robust online simple tracking. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 723–730.

[Schölkopf et al., 1999] Schölkopf, B., Williamson, R. C., Smola, A. J., Shawe-Taylor, J., and Platt, J. C. (1999). Support vector method for novelty detection. In *NIPS*, pages 582–588.

[Sikora, 1997] Sikora, T. (1997). The mpeg-4 video standard verification model. *Circuits and Systems for Video Technology, IEEE Transactions on*, 7(1):19–31.

[Specht, 1991] Specht, D. F. (1991). A general regression neural network. *Neural Networks, IEEE Transactions on*, 2(6):568–576.

[Stauffer, 1999] Stauffer, C. (1999). Automatic hierarchical classification using time-based co-occurrences. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on.*, volume 2, pages 333–339.

[Stauffer and Grimson, 1999] Stauffer, C. and Grimson, W. (1999). Adaptive background mixture models for real-time tracking. In *Computer Vision and Pattern Recognition. IEEE Computer Society Conference on.*, volume 2, pages 246 – 252.

[Stauffer and Grimson, 2000] Stauffer, C. and Grimson, W. E. L. (2000). Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):747–757.

[Steffens et al., 1998] Steffens, J., Elagin, E., and Neven, H. (1998). Personspotter-fast and robust system for human detection, tracking and recognition. In *Automatic Face and Gesture Recognition. Third IEEE International Conference on*, pages 516 –521.

[Sumpter and Bulpitt, 2000] Sumpter, N. and Bulpitt, A. (2000). Learning spatio-temporal patterns for predicting object behaviour. *Image and Vision Computing*, 18(9):697 – 704.

[Sun et al., 2011] Sun, X., Yao, H., and Zhang, S. (2011). A novel supervised level set method for non-rigid object tracking. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 3393–3400.

[Sun et al., 2006] Sun, Z., Bebis, G., and Miller, R. (2006). On-road vehicle detection: a review. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 28(5):694–711.

[Tai et al., 2004] Tai, J.-C., Tseng, S.-T., Lin, C.-P., and Song, K.-T. (2004). Real-time image tracking for automatic traffic monitoring and enforcement applications. *Image and Vision Computing*, 22(6):485 – 501.

[Tao et al., 2002] Tao, H., Sawhney, H., and Kumar, R. (2002). Object tracking with bayesian estimation of dynamic layer representations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 24(1):75 –89.

[Torsello et al., 2006] Torsello, A., Bulò, S. R., and Pelillo, M. (2006). Grouping with asymmetric affinities: A game-theoretic perspective. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages 292–299.

[Tsai and Lai, 2009] Tsai, D.-M. and Lai, S.-C. (2009). Independent component analysis-based background subtraction for indoor surveillance. *Image Processing, IEEE Transactions on*, 18(1):158 –167.

[Tseng and Chang, 1992] Tseng, D. C. and Chang, C. H. (1992). Color segmentation using perceptual attributes. In *Pattern Recognition. Conference C: Image, Speech and Signal Analysis, Proceedings., 11th IAPR International Conference on*, volume 3, pages 228–231.

[Vascon et al., 2013] Vascon, S., Cristani, M., Pelillo, M., and Murino, V. (2013). In *International Conference on Image Analysis and Processing*, volume 8157 of *Lecture Notes in Computer Science*.

[Veenman et al., 2001] Veenman, C., Reinders, M. J. T., and Backer, E. (2001). Resolving motion correspondence for densely moving points. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 23(1):54–72.

[Viola et al., 2003] Viola, P., Jones, M., and Snow, D. (2003). Detecting pedestrians using patterns of motion and appearance. In *Computer Vision. Proceedings. Ninth IEEE International Conference on*, volume 2, pages 734 –741.

[Viola et al., 2005] Viola, P., Jones, M. J., and Snow, D. (2005). Detecting pedestrians using patterns of motion and appearance. *International Journal of Computer Vision*, 63:153–161.

[Wang et al., 2003] Wang, L., Hu, W., and Tan, T. (2003). Recent developments in human motion analysis. *Pattern Recognition*, 36(3):585–601.

[Watada et al., 2010] Watada, J., Musa, Z., Jain, L., and Fulcher, J. (2010). *Human Tracking: A State-of-Art Survey*, volume 6277 of *Knowledge-Based and Intelligent Information and Engineering Systems*, pages 454–463. Springer Berlin / Heidelberg.

[Wei et al., 2007] Wei, Q., Hu, W., Zhang, X., and Luo, G. (2007). Dominant sets-based action recognition using image sequence matching. In *Image Processing, IEEE International Conference on*, volume 6, pages 133–136.

[Wilson et al., 1997] Wilson, A., Bobick, A., and Cassell, J. (1997). Temporal classification of natural gesture and application to video coding. In *Computer Vision and Pattern Recognition. Proceedings. IEEE Computer Society Conference on*, pages 948 –954.

[Wren et al., 1996] Wren, C., Azarbayejani, A., Darrell, T., and Pentland, A. (1996). Pfinder: Real-time tracking of the human body. In Tescher, A. G. and Bove, V. M., editor, *Intefration issues in Large Commercial Media Delivery Systems,. Proceedings of the Society of Photo-Optical Instrumentation Engineers (SPIE)*, volume 2615, pages 89–98.

[Wu and Nevatia, 2007] Wu, B. and Nevatia, R. (2007). Detection and tracking of multiple, partially occluded humans by Bayesian combination of edgelet based part detectors. *International Journal of Computer Vision*, 75(2):247–266.

[Wu and Trivedi, 2005] Wu, J. and Trivedi, M. (2005). Performance characterization for gaussian mixture model based motion detection algorithms. In *Image Processing. IEEE International Conference on*, volume 1, pages 1097–1101.

[Xiang and Gong, 2005] Xiang, T. and Gong, S. (2005). Video behaviour profiling and abnormality detection without manual labelling. In *Computer Vision, IEEE International Conference on*, volume 2, pages 1238–1245.

[Xiang and Gong, 2008] Xiang, T. and Gong, S. (2008). Video behavior profiling for anomaly detection. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(5):893–908.

[Xie and Beni, 1991] Xie, X. L. and Beni, G. (1991). A validity measure for fuzzy clustering. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 13(8):841–847.

[Yang and Ahuja, 1998] Yang, M.-H. and Ahuja, N. (1998). Extraction and classification of visual motion patterns for hand gesture recognition. In *Computer Vision and Pattern Recognition. Proceedings. IEEE Computer Society Conference on*, pages 892–897.

[Yang et al., 2005] Yang, T., Pan, Q., Li, J., and Li, S. (2005). Real-time multiple objects tracking with occlusion handling in dynamic scenes. In *Computer Vision and Pattern Recognition, IEEE Computer Society Conference on*, volume 1, pages 970–975.

[Yang and Latecki, 2011] Yang, X. and Latecki, L. J. (2011). Affinity learning on a tensor product graph with applications to shape and image retrieval. In *Computer Vision and Pattern Recognition, IEEE Conference on*, pages 2369–2376.

[Yilmaz et al., 2006] Yilmaz, A., Javed, O., and Shah, M. (2006). Object tracking: A survey. *ACM Comput. Surv.*, 38(4).

[Yu et al., 2007] Yu, T., Zhang, C., Cohen, M., Rui, Y., and Wu, Y. (2007). Monocular video foreground/background segmentation by tracking spatial-color gaussian mixture models. In *Motion and Video Computing. IEEE Workshop on*, pages 5–13.

[Yu et al., 2011] Yu, X., Chen, X., and Zhang, H. (2011). Accurate motion detection in dynamic scenes based on ego-motion estimation and optical flow segmentation combined method. In *Photonics and Optoelectronics (SOPO), Symposium on*, pages 1 –4.

[Zhan et al., 2008] Zhan, B., Monekosso, D. N., Remagnino, P., Velastin, S. A., and Xu, L.-Q. (2008). Crowd analysis: a survey. *Machine Vision and Applications*, 19(5-6):345 – 357.

[Zhan et al., 2007] Zhan, C., Duan, X., Xu, S., Song, Z., and Luo, M. (2007). An improved moving object detection algorithm based on frame difference and edge detection. In *Image and Graphics. International Conference on*, pages 519 –523.

[Zhang et al., 2009] Zhang, S., Yao, H., and Liu, S. (2009). Dynamic background subtraction based on local dependency histogram. *International Journal of Pattern Recognition and Artificial Intelligence*, 23(7):1397–1419.

[Zhang and Hancock, 2011] Zhang, Z. and Hancock, E. (2011). A graph-based approach to feature selection. In Jiang, X., Ferrer, M., and Torsello, A., editors, *Graph-Based*

Representations in Pattern Recognition, volume 6658 of *Lecture Notes in Computer Science*, pages 205–214. Springer Berlin Heidelberg.

[Zhao et al., 2011] Zhao, J., Xu, Y., Yang, X., and Yan, Q. (2011). Crowd instability analysis using velocity-field based social force model. In *IEEE Visual Communications and Image Processing*, pages 1–4.

[Zhao and Nevatia, 2004] Zhao, T. and Nevatia, R. (2004). Tracking multiple humans in complex situations. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(9):1208 –1221.

[Zivkovic, 2004] Zivkovic, Z. (2004). Improved adaptive gaussian mixture model for background subtraction. In *Pattern Recognition, Proceedings of the 17th International Conference on*, volume 2, pages 28–31 Vol.2.

[Zivkovic and Van Der Heijden, 2004] Zivkovic, Z. and Van Der Heijden, F. (2004). Recursive unsupervised learning of finite mixture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 26(5):651–656.

Appendix A

Evaluation methods

Several different methods are commonly used to evaluate the performance of a system in charge of correctly labeling several elements. In motion detection, the models classify each pixel of each frame as background or foreground. Let FN be the number of foreground pixels incorrectly classified as background, FP the number of background pixels incorrectly classified as foreground, TN the number of correctly classified background pixels, TP the number of correctly classified foreground pixels and $N = FN+FP+TN+TP$ the total number of pixels analyzed. The usual performance metrics are calculated as follows:

$$\begin{aligned} FPR &= \frac{FP}{FP+TN} \\ FNR &= \frac{FN}{FN+TP} \\ Recall &= \frac{TP}{TP+FN} \\ Specificity &= \frac{TN}{TN+FP} \\ PWC &= 100 * \frac{FN+FP}{N} \\ Precision &= \frac{TP}{TP+FP} \end{aligned} \tag{A.1}$$

Appendix B

The Hellinger distance between two pairs of histograms

Let h^b and h^o be two normalized histograms composed of $n = n_C + n_h$ bins. Let h^b_C and h^o_C be two histograms having n_C bins and h^b_G and h^o_G be two histograms having n_G, such that:

$$h^j(i) = \begin{cases} h^j_C(i) & \text{if } i \leq n_C \\ h^j_G(i - n_C) & \text{otherwise} \end{cases} \qquad \forall\, j = o, b \tag{B.1}$$

On the one hand, following (3.1), the Bhattacharyya coefficients BC_C, between h^b_C and h^o_C, and BC_G, between h^b_G and h^o_G, are:

$$\begin{aligned} BC_C &= \sum_{k=1}^{n_C} \sqrt{h^b_C(k) h^o_C(k)} \\ BC_G &= \sum_{k=1}^{n_G} \sqrt{h^b_G(k) h^o_G(k)} \end{aligned} \tag{B.2}$$

On the other hand, following (3.2), the distance d, between h^b and h^o is:

$$d = \sqrt{1 - BC(h^b, h^o)} \tag{B.3}$$

which is transformed, following (3.1), to:

$$d = \sqrt{1 - \sum_{k=1}^{n} \sqrt{h^b(k) h^o(k)}} \tag{B.4}$$

Using (B.1), it can be then decomposed in:

$$\begin{aligned} d &= \sqrt{1-\left(\sum_{k=1}^{n_C}\sqrt{h_C^b(k)h_C^o(k)}+\sum_{k=1}^{n_G}\sqrt{h_G^b(k)h_G^o(k)}\right)} \\ &= \sqrt{\left(1-\sum_{k=1}^{n_C}\sqrt{h_C^b(k)h_C^o(k)}\right)+\left(1-\sum_{k=1}^{n_G}\sqrt{h_G^b(k)h_G^o(k)}\right)-1} \end{aligned} \tag{B.5}$$

And finally, through (B.2), Equation (3.3) can be obtained:

$$d = \sqrt{BC_C + BC_G - 1} \tag{B.6}$$

Appendix C

Comparison of results with crowded scenes abnormal behavior detectors

Although the presented beahvior detection method based on Dominant Sets is not focused on solving crowded scene abnormalities, an Equal Error Rate (EER) comparison with other abnormal behavior detection models in crowded scenes is shown in Table C.1. The UCSD dataset is used as the comparison dataset. The algorithms used as comparison are Cong Sparse method [Cong et al., 2011, Cong et al., 2013], Mahadevan MDT [Mahadevan et al., 2010, Li et al., 2013], Adam method [Adam et al., 2008], Mehran social force [Mehran et al., 2009] and Kim space-time MRF [Kim and Grauman, 2009].

Table C.1. EER (in %) for crowded scenes abnormal behavior detectors

DS	Sparse	MDT	Adam	Social force	Space-time MRF
26	20	25	38	31	40

It can be seen that the method, even though it is not modeled to work in crowded scenes, performs similarly to or better than most of the current state-of-the-art methods. A post-process smoothing, consisting in eliminating isolated positive or negative frames, could be performed to obtain even better results. Generally, when a frame is considered to contain abnormal behavior while surrounded by normal behavior frames, it may be considered as normal, and vice versa.